НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ
Кафедра автоматизованих систем обробки інформації і управління

УДК: 657.371.1

«До захисту допущено»
В.о. завідувача кафедри
_____ *О.А.Павлов*
(підпис) (ініціали, прізвище)
“___” _____2019 р.

Дипломний проект
на здобуття ступеня бакалавра

з напряму підготовки *6.050101 «Комп’ютерні науки»*

на тему: «*Комплекс задач обліку продукції з використанням NFC-міток*»

Виконав: студент *4* курсу, групи *ІС-51*
Мигаль Дмитро Степанович _____
(прізвище, ім’я, по батькові) (підпис)

Керівник *доц., к.т.н., доц. Гриша О.В.* _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант з графічної документації *доц., к.т.н., доц. Тєлишева Т.О.* _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент *доц. каф. ТК, к.т.н., доц. Пасько В.П.* _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студент Мигаль Д.С. _____
(підпис)

Київ – 2019 року

АНОТАЦІЯ

Структура та обсяг роботи. Пояснювальна записка дипломного проекту складається з шести розділів, містить 42 рисунки, 10 таблиць, 1 додаток, 11 джерел.

Дипломний проект присвячений розробці комплексу задач обліку продукції з використанням NFC-міток.

У розділі з інформаційного забезпечення були визначені вхідні та вихідні дані до комплексу задач, була розроблена структура бази даних, яка відповідає поставленим цілям проекту.

Розділ математичного забезпечення присвячений поставленню та обґрунтуванню задач, які необхідні для функціонування нашої системи. Також в ньому були розглянуті методи за допомогою, яких ми будемо вирішувати поставлені задачі

У розділі з програмного забезпечення описані основні засоби розробки комплексу задач, висунуті вимоги до технічного забезпечення, обрано та обґрунтовано архітектуру програмного забезпечення.

У технологічному розділі описана інструкція користувача та проведене тестування комплексу задач.

ОБЛІК ПРОДУКЦІЇ, ІНВЕНТАРИЗАЦІЯ, NEAR FEALD COMMUNICATION, NFC-МІТКА

ABSTRACT

Structure and scope of work.

The explanatory note of the diploma project consists of six sections and contains 41 pictures, 10 tables, 1 attachment, 11 sources.

The diploma project is devoted to the development of a software solution of inventory with the help of NFC-tags.

In the section of information provision input and output data for our set of tasks were identified, a database structure, which corresponds to the goals of the project, was developed.

The section of mathematical support is devoted to the delivery and substantiation of the tasks necessary for the functioning of our system. In addition, the methods by which we will solve the tasks set were considered in this section.

The software section describes the main tools for developing a set of tasks, the requirements for technical support, and chooses and justifies the software architecture.

The technology section describes the user's manual and tests for our set of tasks.

PRODUCTS ACCOUNTING, INVENTORY, NEAR FEALD COMMUNICATION, NFC-TAG

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ ТА ТЕРМІНІВ

БД	База даних
ОС	Операційна система
EMV	(Europay, MasterCard та VISA) Міжнародний стандарт по здійсненню операцій для банківських карток із чипом
MD5	(Message Digest 5) Алгоритм хешування
NFC	(Near Field Communication) Зв'язок на невеликих відстанях
RFID	(Radio frequency identification) Радіочастотна ідентифікація

Brute force – метод рішення криптографічної задачі шляхом перебору всіх можливих варіантів ключа.

Хешування – перетворення вхідного масиву даних довільної довжини у вихідний бітовий рядок фіксованої довжини.

ВСТУП

Дипломний проект присвячений розробці комплексу задач обліку продукції з використанням NFC-міток.

Як відомо, на виробництві часто зіштовхуються з проблемою інвентаризації виготовленої продукції. Зі збільшенням масштабів виробництва зростає і масштаб цієї проблеми, оскільки кількість виготовлених товарів стає важко відслідковувати.

До 1952 року не існувало жодної реальної автоматизованої системи моніторингу, і такі завдання, як підрахунок продукції, запасів чи товарів здійснювались повністю вручну. Це викликало певні труднощі, адже це не лише вимагає багато часу та роботи, а часто надавало неточні результати, які могли вплинути на прибутки компанії. Так було аж до 1970-х років, коли штрих-коди та їх сканери вперше з'явилися у доступі та революціонізували роздрібну торгівлю та дистрибуцію [1].

У 1994 році, приблизно через 20 років після того, як початкові сканери штрих-кодів були введені, стали очевидними обмеження їх використання. Найбільш незручним було те, що сканер міг читати лише 20 буквено-цифрових символів, які б міг зберігати штрих-код. Саме так розпочалась розробка нової системи 2D-коду для обробки – так званих QR-кодів (англ. Quick Response – швидкий відгук). Їх можна сканувати в цифровому форматі, найчастіше за допомогою смартфона. Камера телефону зчитує код, який потім інтерпретується процесором системи. Вона складається з чотирьох квадратів, причому перші три великі діють як цілі для вирівнювання, а менша - нормалізує кут і розмір знімку. Кодовані дані інтерпретуються одним з чотирьох основних режимів: буквено-цифровий, числовий, двійковий чи бітовий, Канзі (ієрогліфічне письмо, частина японської писемності) [1].

З розвитком цієї технології, QR-сканери могли читати більше інформації, охоплюючи площу до 177 пікселів, яка може містити близько 1852 символів. Сьогодні, аналогічно до призначення штрих-кодів, QR-коди

використовуються в доволі багатьох ситуаціях, включаючи інвентаризацію, відстеження запасів, логістику та доставку.

Незважаючи на універсальність і корисність QR-кодів, вони мають декілька недоліків, перший з яких – це труднощі сканування. Швидкість сканування QR-кодів збільшилася більш ніж на третину, з того часу як вони стали популярними як маркетингові та рекламні інструменти в 2010-му – 2013-му роках, але вони як і раніше, як правило, вимагають стійкості руки, безперешкодний огляд для камери, і достатньо часу для їх сканування. Це означає, що QR-коди повинні бути розміщені в місці, де сканування практичне [2].

На даний момент існують інші способи зчитування даних, які є більш прийнятними та зручнішими для користувачів. Одним з них є Near Field Communication – технологія бездротового високочастотного зв'язку малого радіусу дії «в один дотик». Сотні моделей смартфонів, а також планшетів чи телефонів з функціональними можливостями мають вбудовані NFC-зчитувачі, яку використовують багато програм, наприклад Android Pay і Apple Pay, переважно для здійснення безконтактних платежів.

NFC може використовуватися багатьма з тих самих способів, для яких використовують QR-коди, наприклад, надання інформації про продукт, зокрема для інвентаризації [3].

Метою створення комплексу задач є полегшення процесу обліку продукції на виробництві, а також зменшення витрат підприємства на відшкодування коштів за дефектну продукцію, що базується на зібраній статистиці про етапи виробництва, за рахунок превентивних заходів стосовно технологів. Саме завдяки NFC-міток це і буде реалізовано.

Практичне значення одержаних результатів. Розроблено систему обліку продукції на виробництві за допомогою технології NFC.

1 ЗАГАЛЬНІ ПОЛОЖЕННЯ

1.1 Опис предметного середовища

Будь-який продукт проходить безліч етапів перед тим, як дістатись до потенційного користувача чи власника. Зазвичай ми знаємо лише марку виробника, та магазин в якому ми цей товар придбали, тобто лише дві передостанні ланки в цьому ланцюжку. Проте часто цього буває недостатньо як і для поставника, так і для покупця.

Проблеми які виникають у виробника:

1. Якщо під час виробництва дефектний продукт не був вилучений і досягнув кінцевого користувача, який і виявив недоліки, неможливо визначити, які технологи відповідальні за цю неполадку;
2. Неможливість вести статистику по переміщенню продукції на виробництві автоматично.

Проблеми які виникають в покупця:

1. Покупцю не завжди може бути відомо чи продукт був виготовлений оригінальним постачальником;
2. Немає можливості перевірити чи мав товар попередніх власників, чи він є новим;
3. При втраті продукту важко визначити справжнього покупця чи власника, якщо відсутній чек, чи гарантійний талон.

Тобто задача зводиться до створення системи обліку, яка у базі даних зберігатиме інформацію про кожен етап розробки певного продукту на шляху від виробника до користувача.

Near Field Communication (NFC) – з англійської «зв'язок на невеликих відстанях» – технологія що забезпечує бездротовий високочастотний зв'язок малого радіусу дії «в один дотик». Вона надає можливість обмінюватись даними між пристроями, переважно смартфонами та безконтактними платіжними терміналами, що знаходяться на відстані не більше 10 см.

Технологія NFC насправді є розширенням EMV, що дозволяє застосовувати їх для здійснення безконтактних фінансових операцій з платіжним терміналом. Пристрій, наприклад смартфон, із вбудованим NFC-модулем може підтримувати зв'язок як із смарт-картками, так і з терміналами стандарту «ISO 14443», а також з іншими девайсами [4].

Дана технологія може використовуватися на вже існуючій інфраструктурі для безконтактних карток, у громадському транспорті тощо. Передусім NFC розроблена для застосування в мобільних телефонах.

Перевагою цієї технології є швидкість взаємодії між пристроями – близько 0,1 секунди. NFC працює на частоті 13,56 МГц, а швидкість обміну даними становить до 424 кбіт/с.

Для нашої системи найбільше підходять NFC-мітки, на які записують певну послідовність інформації. Здебільшого мітки перебувають в пасивному стані й не споживають енергії. Коли до мітки підносять активний NFC-зчитувач, вона активується й передає інформацію. Здебільшого мітки доступні лише для зчитування.

Кожен продукт на початку виробництва отримує свою NFC-мітку, яка буде розміщена на самому продукті, в доступному для покупця чи технолога місці. На ній буде записаний унікальний ідентифікатор даного продукту, який буде призначений йому при створенні і збережений в базі даних системи.

У ході виробництва на кожному етапі, певний технолог записуватиме дані про те, хто здійснив, та яка саме операція буда здійснена над цим продуктом, прикладаючи телефон з вбудованим NFC-чипом до мітки.

Також на етапі продажу може бути записана інформація про нового власника, а якщо ж він не хоче цього робити, то виробник позначає товар проданим, щоб уникнути перепродажів та крадіжок.

1.1.1 Опис процесу діяльності

Розглянемо послідовність дій які має виконати покупець для перегляду інформації про продукт за допомогою структурної схеми діяльності, представленої на рисунку 1.1.

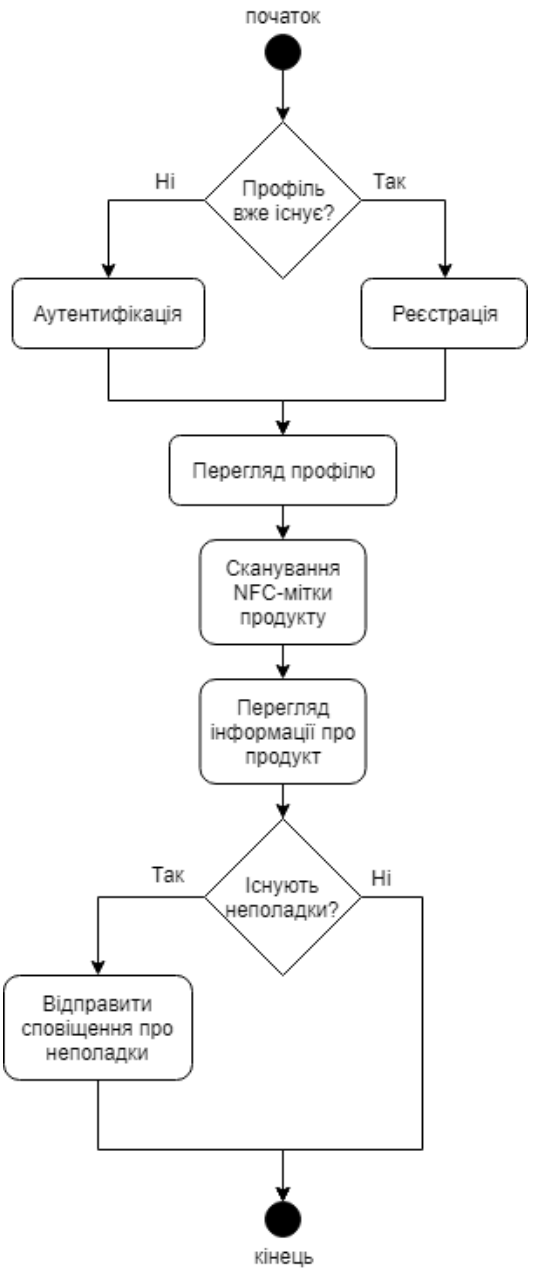

Рисунок 1.1 – Схема структурна діяльності покупця

Розглянемо послідовність дій які має виконати технолог для перегляду та запису інформації про продукт за допомогою структурної схеми діяльності, представленої на рисунку 1.2.

Рисунок 1.2 – Схема структурна діяльності технолога

Крім того, щоб вводити свої дані на сторінці авторизації, технолог має можливість провести автентифікацію за допомогою NFC-мітки, на якій записаний логін та хеш паролю від обліковий запис, що значно спрощує цей процес. Таку ж саму можливість має і адміністратор системи.

Розглянемо послідовність дій які має виконати адміністратор для перегляду та реагування на неполадки в продукції, про які сповістили покупці, а також налаштування виробництва за допомогою структурних схем діяльності, представлених на рисунку 1.3.

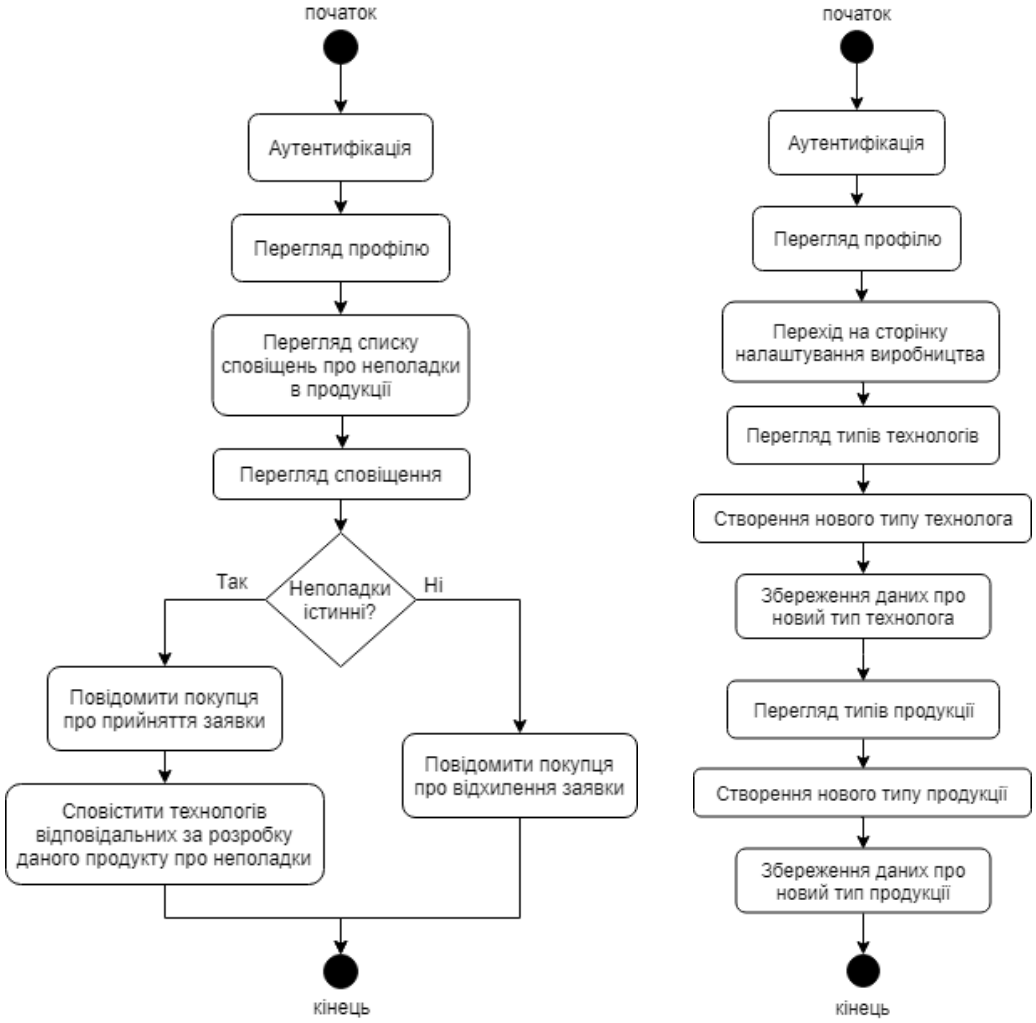

Рисунок 1.3 – Схеми структурні діяльності адміністратора для процесів перегляду та реагування на неполадки і налаштування виробництва

Розглянемо послідовність дій, які має виконати адміністратор для створення нових облікових записів технологів та нових записів про продукції за допомогою структурних схем діяльності, представлених на рисунку 1.4.

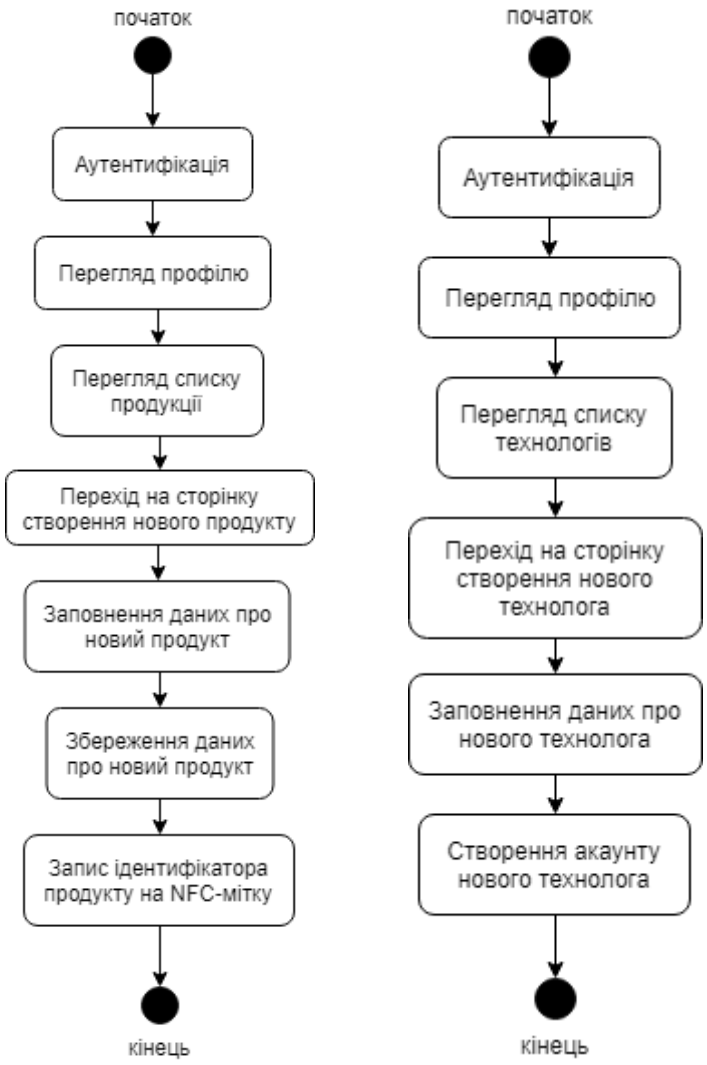

Рисунок 1.4 – Схеми структурні діяльності адміністратора для процесів створення нових технологів та продуктів

1.1.2 Опис функціональної моделі

В даній системі ми можемо визначити три дійові особи (актори):

- адміністратор;
- технолог;
- покупець.

Розглянемо детальніше кожного з акторів, та дії, які вони можуть виконувати всередині системи.

Адміністратор. Працівник на виробництві з найвищим правом доступу до даних, що зберігаються в системі. Даний актор має можливість створювати нових, а також редагувати дані вже існуючих технологів. Також саме він має право створювати новий продукт та записувати його унікальний ідентифікатор на NFC-мітку. Крім того саме цей актор переглядає та реагує на сповіщення покупців про неполадки в продукції. Адміністратор має можливість доступу до даних про всіх технологів, покупців, а також всієї виготовленої продукції.

Технолог. Працівник на виробництві, який має право зчитувати унікальний ідентифікатор продукту з NFC-мітки, і після цього дописувати інформацію про здійснені операції над ним у базу даних системи. Також цей актор має право переглядати дані про всю продукцію, з якою він працював, а також дані про технологів, які працювали з нею на попередніх етапах.

Покупець. Актор який не належить до виробництва, і має доступ лише до обмеженої інформації про продукт. Він може зчитувати дані з NFC-мітки, проте немає права записувати ніяких даних, окрім персональної інформації у базу даних системи. Також покупець може залишити сповіщення про те, що продукт є бракованим, чи містить певні неполадки.

Тепер необхідно визначити дії, які можуть виконувати актори в рамках даної системи у вигляді схем структурних варіантів використання, зображених на рисунках 1.5, 1.6, 1.7.

Рисунок 1.5 – Схема структурна варіантів використання для адміністратора

Рисунок 1.6 – Схема структурна варіантів використання для технолога

Рисунок 1.7 – Схема структурна варіантів використання для покупця

1.2 Огляд наявних аналогів

На сьогоднішній день існує багато систем, які допомагають контролювати інвентаризацію, і відслідковувати місцезнаходження товару на виробництві в даний момент часу. Проте більшість таких систем використовує зчитувачі бар-кодів, які є доволі дорогими, і враховуючи масштаби виробництва, не вельми зручними у використанні.

До того ж, як тільки товар покидає фабрику чи виробничий цех, новий власник товару знає лише торгову марку виробника і не більше, а вся інформація про те, в яких відділах та якими працівниками він був виготовлений, знищується за відсутністю необхідності в ній. Проте насправді ця інформація є більш ніж цінною, адже її можна використовувати і для аналітики і для підвищення контролю за якістю продукції, а також забезпечити клієнта даними про виробництво.

Наразі існує така система, як ShelfAware – розробка компанії O-ring. Вона заснована на технології RFID, що значно спрощує процес інвентаризації, оскільки можна відмовитись від штрих-кодів на виробництві. RFID-мітка розміщується на продукті, а у відділі є 2 зчитувача – по одному на вході та виході, які і реєструють присутність саме цього виробу в даному

приміщенні. Проте, як тільки товар покидає виробничий цех вся інформація про його розробку видаляється [5].

Окрім цього, RFID має доволі великий радіус зчитування даних, що не є безпечно для такої системи, враховуючи те, що ця технологія передбачає сканування одразу всіх міток, які знаходяться в полі доступу. На деяких виробництвах це може призвести до помилок в роботі системи, адже дані, що будуть надходити в неї можуть змішатись, а це буде означати, що жоден з товарів не буде збережений у базу даних [6].

Технологія NFC, яка останнім часом набирає доволі великої популярності, може стати заміною RFID у процесі інвентаризації. Наразі її використовують в багатьох системах та програмних продуктах. Що стосується ринку мобільних пристроїв, то дана технологія найчастіше використовується тільки для двох речей: передача файлів на інший пристрій і оплата покупок та послуг. Саме перша з них стала можлива лише тоді, як Android 4.0 Ice Cream Sandwich був випущений в реліз пошуковою компанією Google, в якому вперше була представлена функція Android Beam. Окрім того, підтримка пристроєм технології Near Field Communication дозволяє йому зчитувати або записувати програмовані NFC-мітки, які можна використовувати для збереження інформації про продукт.

Висновок до розділу

У даному розділі було розглянуто предметне середовище, було визначено та описано процес діяльності, що наведений у структурних схемах діяльності. Також були розглянуті наявні аналоги системи, що буде розроблятись. Крім того, була сформована функціональна модель системи - описані користувачі системи, їх ролі та можливі дії у системі. Функціональна модель представлена у вигляді структурної схеми варіантів використання.

1.3 Постановка задачі

1.3.1 Призначення розробки

Призначенням комплексу задач є розробка системи обліку продукції впродовж всього виробництва за допомогою NFC-міток.

1.3.2 Цілі та задачі розробки

Метою створення комплексу задач є полегшення процесу обліку продукції на виробництві, а також зменшення витрат підприємства на відшкодування коштів за дефектну продукцію, що базується на зібраній статистиці про етапи виробництва, за рахунок превентивних заходів стосовно технологів.

Для досягнення поставленої цілі мають бути вирішені такі задачі:

- облік кожного етапу виготовлення продукції;
- облік зворотного зв’язку від покупців;
- створення програмного застосунку, для можливості сканування та редагування NFC-міток;
- побудова звітів по дефектній продукції;
- шифрування інформації записаної на NFC-мітку;
- шифрування інформації записаної в базу даних системи.

Висновок до розділу

В даному розділі був сформований опис предметного середовища, процесу діяльності, який я автоматизую, функціональної моделі та перегляд наявних аналогів. Також тут була здійснена постановка задачі, а саме призначення, мета та задачі розробки системи. Були наведені структурні схеми діяльності на варіантів використання для всіх акторів системи.

2 ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

2.1 Вхідні дані

Вхідні дані комплексу задач надходять з декількох джерел, а саме від:

- адміністратора;
- технологів;
- покупців.

Тепер детально розглянемо, які саме дані надходять з цих джерел.

Дані, які надходять від адміністратора. Адміністратор – це основне джерело даних для нашої системи. Саме він забезпечує таку ключову інформацію, як:

- види операцій, їх назви, для яких продуктів призначені та максимальні тривалості;
- види технологів, їх назви, та які види операцій вони можуть виконувати;
- види продуктів, та їх назви;
- порядок видів операцій, які проводяться над певним видом продукту;
- логіни, паролі для існуючих технологів, та до якого виду вони належать;
- відповідь на запити покупців про недоліки в придбаній продукції.

Дані, які надходять від технологів. Технологи не мають можливості змінювати дані в системи, вони можуть лише додавати та переглядати записи. Вони поставляють таку інформацію:

- дата, час та тривалість операції, яка була здійснена над певним продуктом.

Дані, які надходять від покупців. Покупці мають можливість лише переглядати записи в системі, які пов’язані з продукцією, яку вони купили. Вони забезпечують таку інформацію:

- логін та пароль для свого аканту;
- персональну інформацію;
- дата, час та розмір оплати покупки певного товару;
- відгук про куплений продукт.

2.2 Вихідні дані

Вихідними даними для покупця в нашій системі є список операцій здійснених над продуктом, який він придбав. Список операцій представляється наступними даними:

- назва технолога та його тип;
- тип операції, дата проведення, тривалість проведення.

Також покупець має можливість переглянути кількість попередніх покупців даного товару. Крім того він може переглянути всі свої запити про недоліки в продукції, що він придбав та відповідь від адміністратора на його скаргу.

Вихідними даними для технологів є список продукції, яку вони опрацювали. Цей список представляється наступними даними:

- тип продукту;
- даті і час проведення операції.

Вихідними даними для адміністратора, є вся інформація про технологів, продукцію, покупців, здійснені технологами операції над продукцією. Також інформація про покупців, та покупки які вони зробили і відгук про придбаний товар.

2.3 Опис структури бази даних

На рисунку 2.1 наведена ER-діаграма для нашої системи.

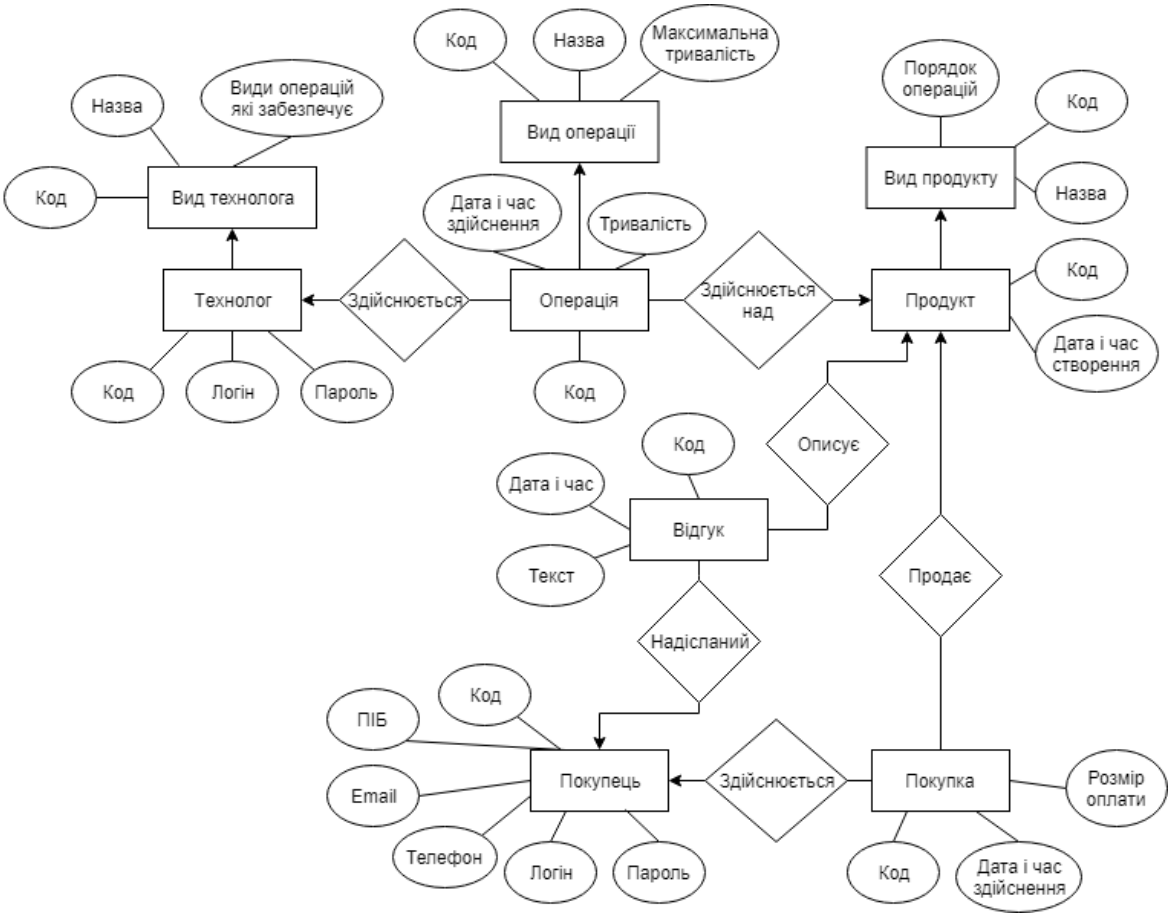

Рисунок 2.1 – ER-діаграма бази даних

Оскільки для нашої системи ми будемо використовувати MongoDB – документо-орієнтовану систему керування базами даних – то структури будуть подані у вигляді ієрархічних структур даних. Документи всередині документо-орієнтованої бази даних у певному розумінні подібні до записів або рядків реляційних баз даних, але вимоги до них не такі жорсткі. Вони не потребують задовольняти стандартній схемі, ані мати однакові секції, частини або ключі. Всі документи подібної одного типу зберігаються в колекції, що є певним аналогом таблиць.

Тепер представимо структури документів для всіх колекцій нашої бази даних на рисунках 2.2 – 2.10.

```
{
   "_id" : ObjectId,
   "name" : String,
   "email" : String,
   "phone" : String,
   "login" : String,
   "pass" : String
}
```

Рисунок 2.2 – Структура документів в колекції покупців Customers

```
{
   "_id" : ObjectId,
   "name" : String
   "duration" : Number
}
```

Рисунок 2.3 – Структура документів в колекції типів операцій OperationTypes

```
{
   "_id" : ObjectId,
   "name" : String,
   "operationsSequence" : [ObjectId]
}
```

Рисунок 2.4 – Структура документів в колекції типів продукції ProductTypes

```
{
   "_id" : ObjectId,
   "name" : String,
   "operations" : [ObjectId]
}
```

Рисунок 2.5 – Структура документів в колекції типів технологів TechnologistTypes

```
{
   "_id" : ObjectId,
   "name" : String,
   "login" : String,
   "pass" : String,
   "technologistType" : ObjectId
}
```

Рисунок 2.6 – Структура документів в колекції технологів Technologists

```
{
   "_id" : ObjectId,
   "productType" : String,
   "datetime" : ISODate
}
```

Рисунок 2.7 – Структура документів в колекції продуктів Products

```
{
   "_id" : ObjectId,
   "operationType" : String,
   "technologist" : ObjectId,
   "product" : ObjectId,
   "datetime" : ISODate,
   "duration" : Number
}
```

Рисунок 2.8 – Структура документів в колекції операцій Operations

```
{
   "_id" : ObjectId,
   "customer" : ObjectId,
   "product" : ObjectId,
   "datetime" : ISODate,
   "price" : Number
}
```

Рисунок 2.9 – Структура документів в колекції покупок Purchases

```
{
   "_id" : ObjectId,
   "customer" : ObjectId,
   "product" : ObjectId,
   "messages" : [{
        "sender" : String,
        "text" : String
   }],
   "result" : Boolean,
}
```

Рисунок 2.10 – Структура документів в колекції відгуків Feedbacks

2.4 Структура масивів інформації

Оскільки для ідентифікації продукції в нашій системі використовується технологія NFC, то на кожній мітці має бути записаний хеш-номер даного товару. Такі мітки мають доволі обмежений діапазон пам’яті – менше одного кілобайта. Саме тому записувати на них ми будемо лише значення, без жодної структури по верху.

Логін та пароль до облікового запису користувача будуть записані на мітку у вигляді JSON об’єкта у вигляді, що зображений на рисунку 2.11.

```
{
   "login" : String,
   "password" : ObjectId
}
```

Рисунок 2.11 – Структура авторизаційних даних користувача

Висновок до розділу

В цьому розділі були сформульовані рішення по інформаційному забезпеченню системи, що проектується, включаючи визначення вхідних та вихідних даних. Також була визначена структура бази даних та масивів інформації. Була наведена ER-діаграма із детальним поясненням до неї.

3 МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1 Змістовна постановка задачі

Будь-який продукт проходить безліч етапів перед тим, як дістатись до потенційного користувача чи власника. Зазвичай ми знаємо лише марку виробника, та магазин в якому ми цей товар придбали, тобто лише дві передостанні ланки в цьому ланцюжку. Проте часто цього буває недостатньо як і для поставника, так і для покупця.

Проблеми які виникають у виробника:

1. Якщо під час виробництва дефектний продукт не був вилучений і досягнув кінцевого користувача, який і виявив недоліки, неможливо визначити, які технологи відповідальні за цю неполадку;
2. Неможливість вести статистику по переміщенню продукції на виробництві автоматично.

Проблеми які виникають в покупця:

1. Покупцю не завжди може бути відомо чи продукт був виготовлений оригінальним постачальником;
2. Немає можливості перевірити чи мав товар попередніх власників, чи він є новим;
3. При втраті продукту важко визначити справжнього покупця чи власника, якщо відсутній чек, чи гарантійний талон.

Тобто задача зводиться до створення системи обліку, яка у базі даних зберігатиме інформацію про кожен етап розробки певного продукту на шляху від виробника до користувача.

Near Field Communication (NFC) – з англійської «зв'язок на невеликих відстанях» – технологія що забезпечує бездротовий високочастотний зв'язок малого радіусу дії «в один дотик». Вона надає можливість обмінюватись даними між пристроями, переважно смартфонами та безконтактними платіжними терміналами, що знаходяться на відстані не більше 10 см.

Технологія NFC насправді є розширенням EMV, що дозволяє застосовувати їх для здійснення безконтактних фінансових операцій з платіжним терміналом. Пристрій, наприклад смартфон, із вбудованим NFC-модулем може підтримувати зв'язок як із смарт-картками, так і з терміналами стандарту «ISO 14443», а також з іншими девайсами [4].

Дана технологія може використовуватися на вже існуючій інфраструктурі для безконтактних карток, у громадському транспорті тощо. Передусім NFC розроблена для застосування в мобільних телефонах.

Перевагою цієї технології є швидкість взаємодії між пристроями – близько 0,1 секунди. NFC працює на частоті 13,56 МГц, а швидкість обміну даними становить до 424 кбіт/с.

Для нашої системи найбільше підходять NFC-мітки, на які записують певну послідовність інформації. Здебільшого мітки перебувають в пасивному стані й не споживають енергії. Коли до мітки підносять активний NFC-зчитувач, вона активується й передає інформацію. Здебільшого мітки доступні лише для зчитування.

Кожен продукт на початку виробництва отримує свою NFC-мітку, яка буде розміщена на самому продукті, в доступному для покупця чи технолога місці. На ній буде записаний унікальний ідентифікатор даного продукту, який буде призначений йому при створенні і збережений в базі даних системи.

У ході виробництва на кожному етапі, певний технолог записуватиме дані про те, хто здійснив, та яка саме операція буда здійснена над цим продуктом, прикладаючи телефон з вбудованим NFC-чипом до мітки.

Також на етапі продажу може бути записана інформація про нового власника, а якщо ж він не хоче цього робити, то виробник позначає товар проданим, щоб уникнути перепродажів та крадіжок.

3.2 Математична постановка задачі

Задача зменшення витрат підприємства на відшкодування коштів за дефектну продукцію

Задача зменшення витрат підприємства на відшкодування коштів за дефектну продукцію за рахунок превентивних заходів стосовно технологів базується на зібраній статистиці про операції здійснені на певних етапи виробництва. В загальному, наше виробництво можна представити у вигляді графу, де наші технологи будуть вузлами, а продукти переміщатимуться між ними, в залежності від встановленого порядку виробництва. Нехай у нас є *n* технологів та *m* операцій, необхідних для виготовлення певного товару. Тоді наш граф можна представити так, як він зображений на рисунку 3.1.

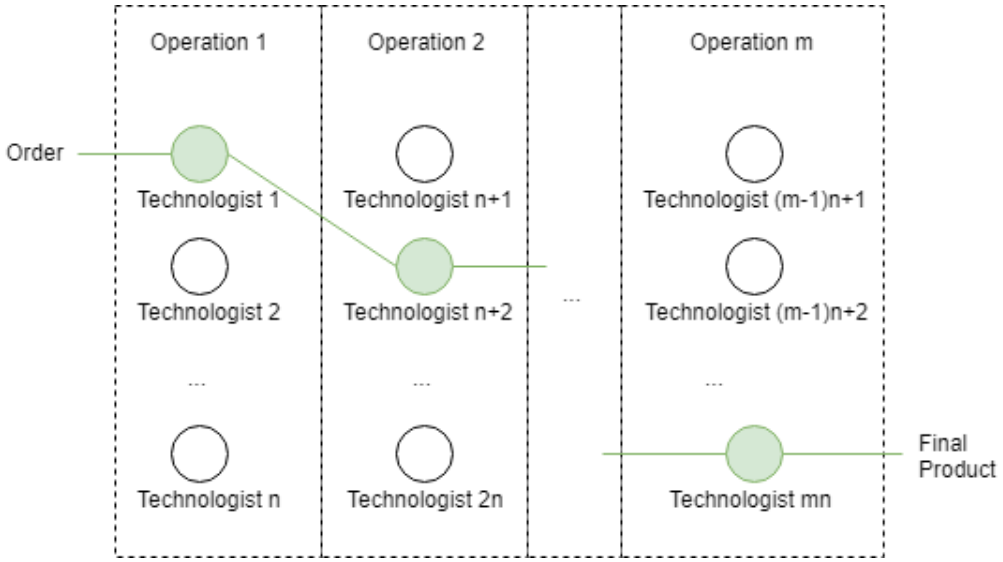

Рисунок 3.1 – Граф, що зображує процес виробництва певного товару

Необхідно визначати, які з вузлів найбільше впливають на кількість виготовленої дефектної продукції, для того, щоб вжити заходи до цих технологів і зменшити витрати на відшкодування.

Задача шифрування інформації записаної на NFC-мітку

Призначенням цієї задачі є мінімізація можливості отримання інформації користувачам, які не мають права доступу до неї. Оскільки давати доступ будь-кому, хто зміг зчитати логін та пароль користувача системи з мітки не є безпечно, необхідно забезпечити захищеність даних.

На вході ми маємо пароль до нашого облікового запису в системі, а на виході ми повинні отримати хеш-номер, який можна зчитати з мітки і який має співпасти з одним з полем «password» в документі з даними про технолога в колекції Technologists нашої бази даних.

3.3 Обґрунтування методу розв'язання

Задача зменшення витрат підприємства на відшкодування коштів за дефектну продукцію

Для того щоб вирішити цю задачу, було вирішено ввести коефіцієнт для кожного вузла, який буде збільшуватись щоразу, як товар, виготовлений даним технологом, буде визначений дефектним. У випадку, коли користувач, або перевірка якості фінального продукту встановить присутність неполадок в певному товарі, то всі технологи, які були залучені до його виробництва отримуватимуть штраф. Такий метод накопичення дозволить чітко відображати картинку ефективності та компетентності працівників на виробництві.

Задача шифрування інформації записаної на NFC-мітку

Для отримання хеш-номеру нам потрібно використати одну з існуючих хеш-функцій. Хеш-функція призначена для згортки вхідного масиву будь-якого розміру в бітовий рядок. Найчастіше хеш-функції використовуються для перевірки унікальності пароля, файлу, рядка і т.д.

Наприклад у вас є два масиви символів, а вам необхідно швидко порівняти їх на рівність, то хеш-функція може зробити це за вас, якщо у двох масивів хеші різні, то вони гарантовано різні, а в разі рівності хешів - масиви швидше за все одинакові.

Для нашого методу ми використаємо алгоритм хешування MD5 розроблений професором Рональдом Л. Ривестом з Массачусетського технологічного інституту (Massachusetts Institute of Technology, MIT) в 1991 році. Для MD5 довжина вихідного рядка дорівнює 128 бітам [7].

MD5 можна вважати безпечним алгоритмом хешування, оскільки ця функція є необоротна і ідентифікатори, збережені у вигляді таких хешів, зламати не можна, навіть якщо зловмисник отримав доступ до бази даних. Проте існують такі методи, як brute force та різноманітні Rainbow-таблиці, які за допомогою перебору підбирають оригінальний масив символів із хеш-номеру. Саме тому необхідно використати такий метод, як сіль. Цей метод передбачає додаткові операції, які необхідно провести над отриманим хеш-номером, для того, мінімізувати можливість отримати вхідний рядок [8].

3.4 Опис методів розв'язання

Задача зменшення витрат підприємства на відшкодування коштів за дефектну продукцію

Нехай n – кількість технологів на виробництві, а r_i – це накопичений штраф i-го технолога, i є [1, n], що на початку дорівнює нулю.

У випадку коли продукт, який дійшов до фінального етапу, був визначений дефектним, то у всіх технологів, які були причетні до його виготовлення, штраф збільшується на одиницю.

$$r_k = r_k + 1,\ k \in A,\ A \subseteq [1, n]$$

На рисунку 3.2 можемо побачити ситуацію, коли був виготовлений бракований продукт.

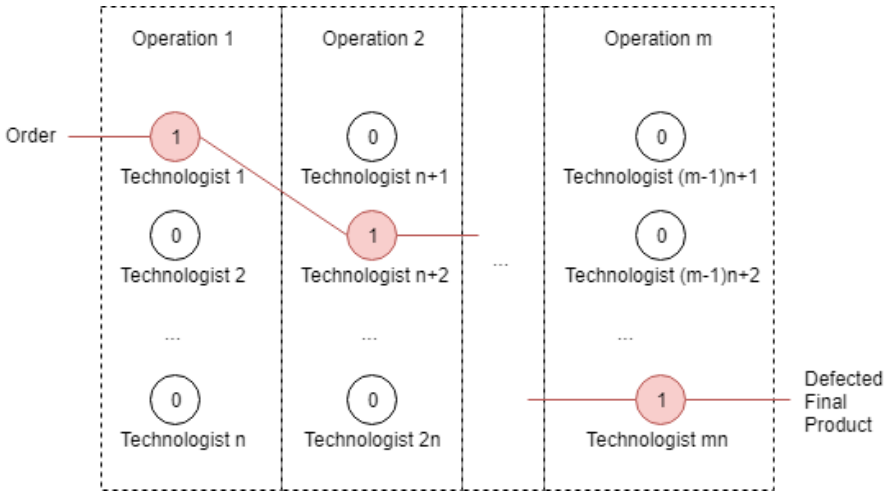

Рисунок 3.2 – Граф, що зображує процес виробництва бракованого товару

З нарощуванням масштабу виробництва, коефіцієнт r в технологів буде зростати, і оскільки шляхи виробництва продукції можуть поєднувати різні вузли, то можна буде визначити, на якому моменті виробництва, тобто в якого технолога, було виготовлено найбільше дефектної продукції. Тобто цільовою функцією цією задачі є мінімізація максимального коефіцієнту r серед усіх технологів.

$$max(r_i) \rightarrow min,\ i \in [1, n]$$

На рисунку 3.3 можна побачити, як виглядатиме граф, якщо 2 дефектних продукти будуть виготовлятись в одного і того ж технолога.

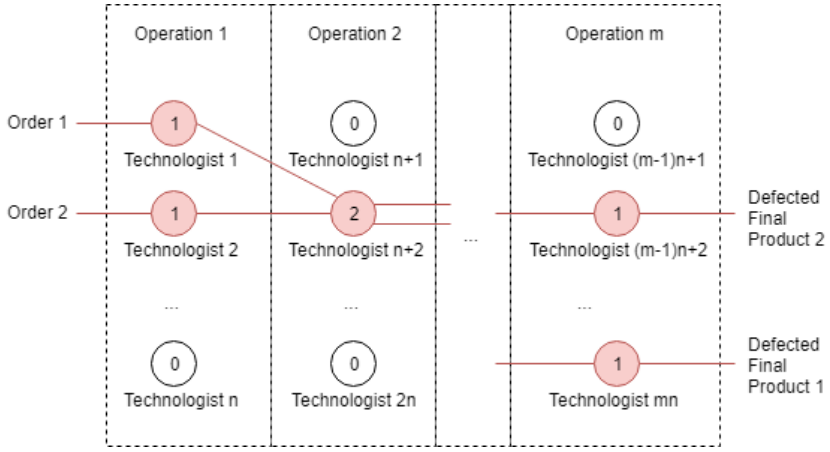

Рисунок 3.3 – Граф, що зображує ситуацію з двома дефектними продуктами

Тобто маємо задачу лінійного програмування у вигляді:

$$max(r_i) \rightarrow min,\ i \in [1, n],$$

$$r_i \geq 0,\ i \in [1, n].$$

Нехай $r_x = max(r_i),\ i \in [1, n]$, тоді адміністратор може здійснити певні заходи щодо технолога x, або ж замінити його, щоб зменшити кількість виготовлення бракованої продукції, що означає зменшення витрат на відшкодування. Тоді коефіцієнт r_x прийме значення 0, і мінімізація буде досягнута. Тобто математично досягнути мінімізації ми не можемо, проте необхідно вказати користувачу, де саме необхідно застосовувати зміни на виробництві. Визначення r_x буде реалізовано за допомогою методу повного перебору елементів, оскільки кількість робітників – це невелике число.

Нижче представлено псевдокод, за допомогою якого можна визначити максимальне значення штрафу, та порядковий номер цього технолога.

```
input: n – кількість технологів;
        r ∈ A – штрафи технологів
output: max – максимальне значення штрафу;
          maxi – індекс технолога з максимальним штрафом

max = 0
i = 0
maxi = 0
for all r ∈ A:
   i = i + 1
   if r > max:
        max = r
        maxi = i
```

Рисунок 3.4 – Псевдокод для визначення максимального значення в масиві

Задача шифрування інформації записаної на NFC-мітку

Хеш-функція MD5 складається з п’яти етапів:

1. Append Padding Bits

У вихідний рядок дописують одиничний байт 0x80, а потім дописують нульові біти, до тих пір, поки довжина повідомлення не буде рівна 448 по модулю 512. Тобто дописуємо нулі до тих пір, поки довжина нового повідомлення не дорівнюватиме (512 * N + 448), де N - будь-яке натуральне число, таке, щоб даний вислів був найближчий до довжини блоку.

2. Append Length

Далі в повідомлення дописується 64-бітове представлення довжини вихідного повідомлення (кількість біт в повідомленні). Після цього довжина потоку стане кратна 512. Обчислення будуть ґрунтуватися на представленні цього потоку даних у вигляді масиву слів по 512 біт.

3. Initialize MD Buffer

Для обчислень ініціалізуються 4 змінні розміром по 32 біта і задаються початкові значення шістнадцятирічними числами. У цих змінних будуть зберігатися результати проміжних обчислень. Початковий стан ABCD називається ініціалізуючим вектором.

4. Process Message in 16-Word Blocks

На четвертому кроці в першу чергу визначається 4 допоміжні логічні функції, які перетворять вхідні 32-бітові слова, в 32-бітові вихідні. Для кожного раунду буде потрібна своя функція:

- перший раунд: $F(X,Y,Z) = (X \cap Y) \cup (\bar{X} \cap Z)$;
- другий раунд: $G(X,Y,Z) = (X \cap Z) \cup (\bar{Z} \cap Y)$;
- третій раунд: $H(X,Y,Z) = X \oplus Y \oplus Z$;
- четвертий раунд: $H(X,Y,Z) = Y \oplus (\bar{Z} \cap X)$.

Також на цьому етапі реалізується так званий «білий шум» - посилення алгоритму, що складається 64-елементного масиву, що містить псевдовипадкові числа, залежні від синуса числа і:

$$T[i] = 4294967296 \cdot |\sin(i)|$$

Кожен 512-бітний блок проходить 4 етапи обчислень по 16 раундів. Для цього блок представляється у вигляді масиву X з 16 слів по 32 біта.

Кожен раунд складається з 16 елементарних перетворень, які в загальному вигляді можна представити у вигляді [abcd ksi], яке, в свою чергу, можна уявити як:

A = B + ((A + F (B, C, D) + X [k] + T [i]) <<< s)

Де:

- A, B, C, D – регістри
- F(B, C, D) - одна з логічних функцій
- X [k] - k-тий елемент 16-бітного блоку.
- T [i] - і-тий елемент таблиці «білого шуму»
- <<< s - операція циклічного зсуву на s позицій вліво.

Заносимо в блок даних елемент n з масиву 512-бітних блоків. Зберігаються значення A, B, C і D, що залишилися після операцій над попередніми блоками (або їх початкові значення, якщо блок перший):

$$A = A + AA; B = B + BB; C = C + CC; D = D + DD$$

5. Output

Виводимо побайтово наш буфер ABCD починаючи з A і закінчуючи D отримаємо наш хеш [9].

Суть методу додавання солі до хешу полягає в наступному. При генерації хешу, крім голого значення отриманого з md5, потрібно провести ввести ще декілька операцій над ідентифікатором. Необхідно згенерувати так звану «сіль» - завдяки якому, лише користувачі, які знають його, зможуть отримати доступ до інформації. Новий, «засолений» хеш буде визначатись наступним чином: хеш = md5(ключ + md5(ідентифікатор))

Таким чином ми отримуємо хеш, збільшеної довжини з конкатенацією пароля з тією ж самої сіллю. Зрозуміло, що в такому випадку генерацією Rainbow-таблиць, саме для цієї «солі», ніхто займатися не буде. В такому випадку зламати результат, навіть знаючи ключ, практично неможливо [8].

Висновок до розділу

В даному розділі було розглянуто, поставлено та математично обґрунтовано задачі, які необхідні для функціонування нашої системи. Також були розглянуті методи за допомогою яких ми будемо вирішувати поставлені задачі.

Для задачі зменшення витрат підприємства на відшкодування коштів за дефектну продукцію був обраний метод нараховування штрафів. Метод був детально описаний, а також були прикріплені 2 схематичні рисунки.

Для задачі шифрування інформації записаної на NFC-мітку вирішено використати метод MD5, що забезпечує високу надійність зашифрованих даних. Опис алгоритму та спосіб його застосування теж описаний в цьому розділі.

4 ПРОГРАМНЕ ТА ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ

4.1 Засоби розробки

Оскільки наша система вимагає присутність NFC-зчитувача в пристрої, на якому вона буде запущена, то найкращим виходом буде реалізувати програмний продукт для смартфонів, адже в сучасному світі більшість цих пристроїв використовують технологію NFC. При створенні програмного продукту було вирішено використовувати фреймворк React Native.

React Native - це фреймворк для розробки кроссплатформенних програм для iOS і Android. Він появився на початку 2015 року і побудований на базі ReactJS - відкритої JavaScript бібліотеки для створення інтерфейсів користувача, що розробляється компанією Facebook. Іншими словами: завдяки React Native веб-розробники можуть писати мобільні додатки, які є справді нативними, за допомогою JavaScript бібліотеки. Крім того, оскільки більшість коду, який ви пишете, можна використовувати для різних платформами, React Native спрощує одночасну розробку як для Android, так і для iOS [10].

Подібно до ReactJS, програми React Native написані з використанням суміші JavaScript і XML-подібної розмітки, відомої як JSX. Потім React Native використовує рідні API для візуалізації в Objective-C (для iOS) або Java (для Android). Таким чином, програма відображатиметься за допомогою реальних компонентів мобільного інтерфейсу, а не веб-сторінок, а також виглядатиме як будь-який звичайний мобільний додаток.

React Native підтримує як iOS, так і Android, а також має можливість розширюватися і на платформи, які виникнуть у майбутньому. React Native також надає інтерфейси JavaScript для API платформи, тому програми можуть отримувати доступ до вбудованих функцій, таких як камера телефону або місцезнаходження користувача. Саме завдяки цьому ми можемо працювати з NFC, зчитувати та записувати дані з міток.

Для роботи з NFC, необхідно підключити модуль Node.js, що має назву react-native-nfc-manager. Він дозволяє легко налаштувати використання NFC-модуля в нашому застосунку, а також кодувати і декодувати дані, які записані на мітках. Ще одною перевагою є те, що функції, які реалізовані в цьому модулі є кросплатформеними, тобто працюють як на IOS так і на Android застосунках.

Для реалізації другого та третього рівня нашої системи було вирішено розробити RESTful API. REST - скорочення від Representational State Transfer, що можна перекласти як «передача репрезентативного стану». Це стиль проектування розподілених систем за допомогою обмежень. Центральною абстракцією в REST є ресурс, а головні обмеження виглядають так:

- клієнт-серверна модель;
- взаємодія без збереження стану;
- логічний інтерфейс.

В клієнт-серверній моделі сервер надає якийсь сервіс чи ресурси, котрі отримують клієнти, виконуючи запити, при чому клієнт може бути чим завгодно: Android-додатком, браузером чи банкоматом. Перевага такого принципу в тому, що так підтримується розподіл інтересів - якщо в вас є єдина машина-сервер, цього достатньо, а клієнти можуть бути різними [11].

В архітектурі REST сервер не повинен зберігати ніякої інформації про стан операції, сесії повинен зберігати клієнт. Це означає, що якщо сервер отримав два різних запити від одного клієнта, вони не повинні впливати один на одного. Через це, всю інформацію, потрібну для здійснення дії, клієнт повинен відправляти відразу. Такий підхід дозволяє економити купу часу та ресурсів і полегшує масштабування сервера.

Ресурс - це представлення віртуального об'єкту (такого як зображення), реального об'єкту чи колекції об'єктів. Взагалі, ресурс може бути чим завгодно, розробник API вирішує що в нього буде ресурсом. В REST кожен ресурс повинен бути унікальним, тому ми їм присвоюється ідентифікатор.

Наприклад для нашої мережі /products/1 та /products/2 це два різних продукти з ідентифікаторами 1 та 2. Також ресурси можуть бути вкладеними. Наприклад, URI операцій здійснених над першим продуктом буде виглядати так /products/1/operations.

Більшість запитів до нашого API будуть на виконання базових CRUD-операцій (англ. Create, Read, Update, Delete – створення, читання, оновлення, видалення). Для цього використовуються HTTP методи, такі як GET, POST, PUT / PATCH та DELETE.

Для розробки нашого RESPful API було вирішено використовувати Node.js для серверного застосунку і MongoDB для збереження даних. Node.js - це середовище виконання JavaScript, побудована на JavaScript-двигуні V8 з Chrome. В основі Node.js лежить подієво-керована модель з неблокуючими операціями I / O, що робить її легкою та ефективною. Іншими словами: Node.js дає можливість писати неймовірно продуктивний серверний код з використанням JavaScript. У Node.js використовується libuv - крос-платформна бібліотека підтримки з акцентом на асинхронний input-output.

Також для розробки серверної частини використовується модуль Express.js, або просто Express - фреймворк для Node.js. Він спроектований для створення веб-додатків і API, і де-факто є стандартним фреймворком для Node.js. Автор фреймворка, TJ Holowaychuk, описує його як створений на основі написаного на мові Ruby каркаса Sinatra, маючи на увазі, що він мінімалістичний і включає велику кількість додаткових плагінів.

Для збереження даних була обрана MongoDB - документоорієнтована система керування базами даних (СКБД) з відкритим вихідним кодом, яка не потребує опису схеми таблиць, написана на мові C++. Вона класифікована як NoSQL і використовує JSON-подібні документи і схему бази даних. Оскільки на кожному рівні системи ми використовуємо JavaScript, то JSON це найкращий спосіб передавати дані, адже він легко транслюється в JavaScript об’єкти.

4.2 Вимоги до технічного забезпечення

Для правильної роботи даної програми до складу технічних засобів повинні входити:

Смартфон, для кожного працівника в компанії з такими характеристиками:

- працює на базі однієї з найпопулярніших операційних систем IOS чи Android;
- має вбудований NFC-модуль;
- має достатній об’єм оперативної пам’яті для роботи застосунку;
- має доступ до інтернету.

Крім того, необхідні NFC-мітки з достатнім об’ємом пам’яті, на які будуть записуватись ідентифікатори продукції.

4.2.1 Загальні вимоги

4.2.2 Опис локальної обчислювальної мережі

Система, що розробляється має трирівневу архітектуру. Клієнтом в нашому випадку є мобільний застосунок, який представляє перший рівень призначений для кінцевого користувача. Він не має прямих зв'язків з базою даних, не є навантаженим основною бізнес-логікою і зберігає стан програми.

На другому рівні розташовується сервер застосунків, в нашому випадку це сервер реалізований за допомогою Node.js та зокрема фреймворком Express.js. На другому рівні зосереджена більша частина бізнес-логіки, а поза ним залишаються фрагменти, що експортуються з бази даних.

На третій рівень виноситься Сервер бази даних, що забезпечує зберігання даних. В нашій системі використовується MongoDB - документо-орієнтована система керування базами даних, яка не потребує опису схеми таблиць, а працює з колекціями та документами.

4.3 Архітектура програмного забезпечення

4.3.1 Діаграма класів

Для даної системи не потрібно утворювати класи, оскільки вона є повністю функціональною.

4.3.2 Діаграми послідовності

Рисунок 4.1 – Схема структурна послідовності для прецедентів *Створення нового технолога* та *Створення нового продукту* в системі

Рисунок 4.2 – Схема структурна послідовності для прецеденту *Збереження даних про операцію* в системі обліку продукції

Рисунок 4.3 – Схема структурна послідовності для прецеденту *Реєстрація покупця* в системі обліку продукції

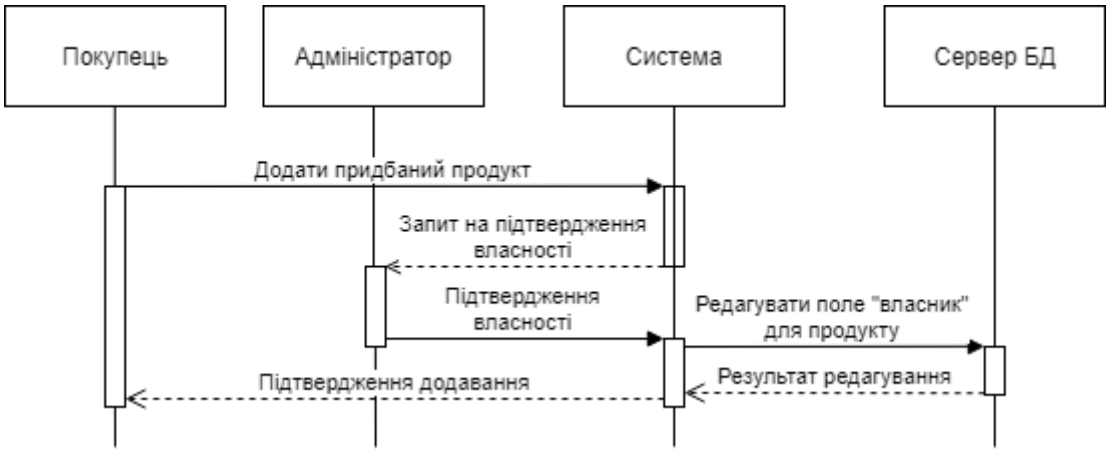

Рисунок 4.4 – Схема структурна послідовності для прецеденту *Покупка продукту* в системі обліку продукції

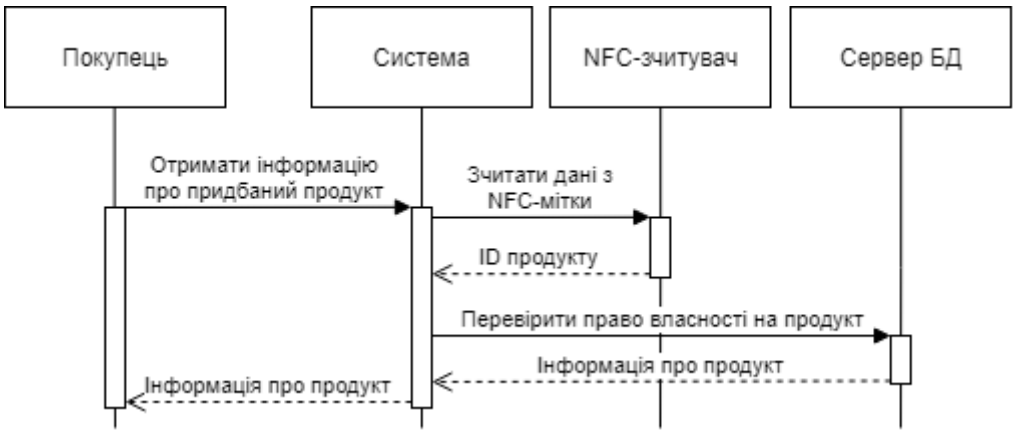

Рисунок 4.5 – Схема структурна послідовності для прецеденту *Зчитування даних продукту з NFC-мітки* в системі обліку продукції

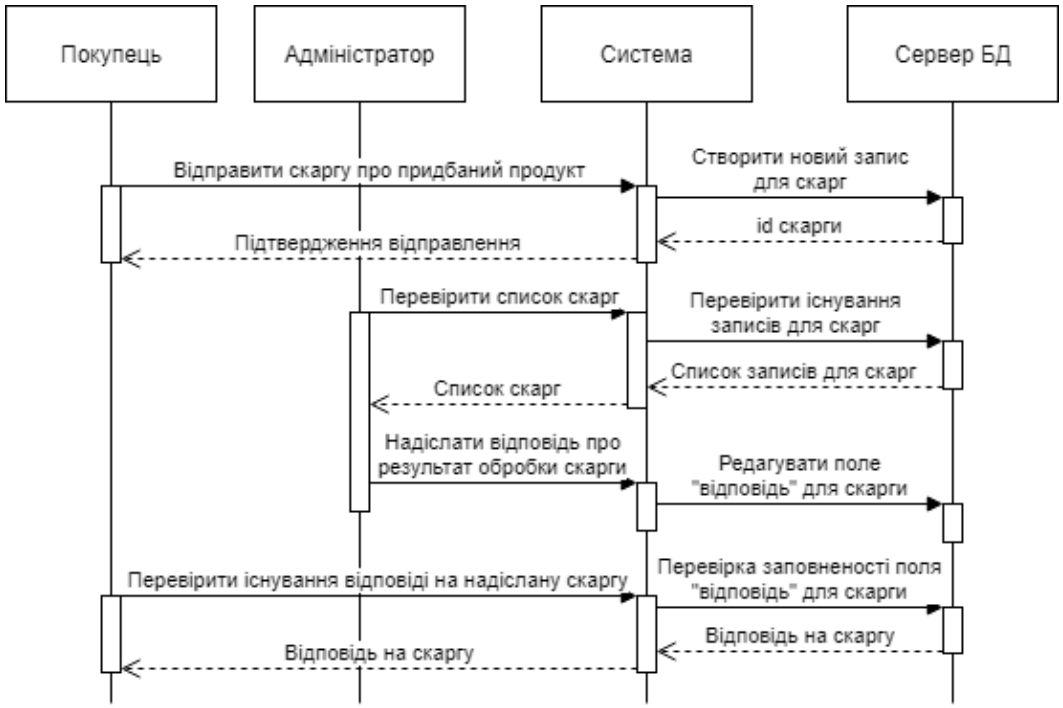

Рисунок 4.6 – Схема структурна послідовності для прецеденту *Обробка скарг на дефектний товар* в системі обліку продукції

4.3.3 Діаграма розгортання

Оскільки компонентів в даній системі доволі багато, представити їх діаграмою компонентів є доволі важко, тому було вирішено реалізувати діаграму розгортання.

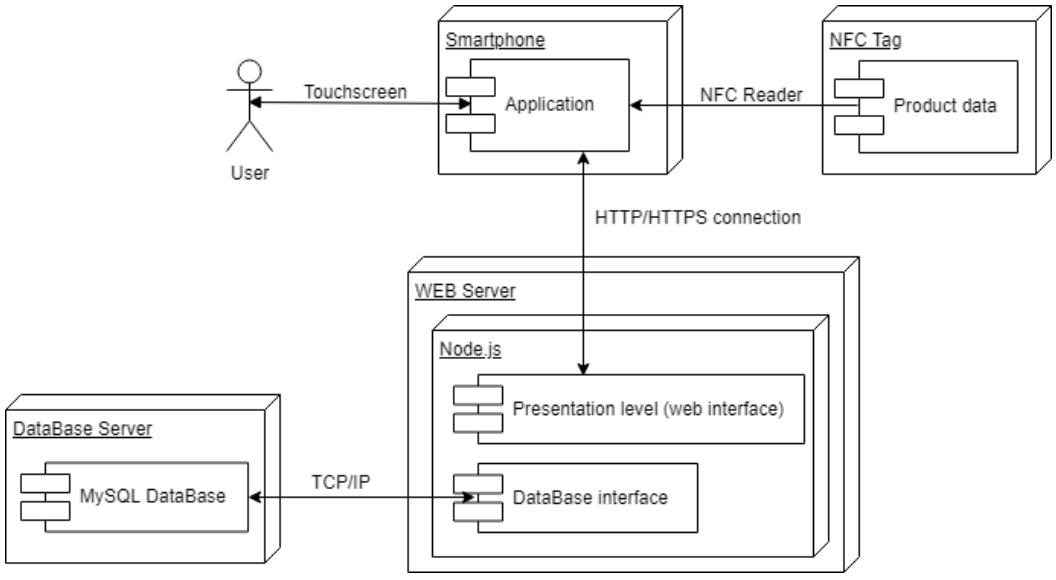

Рисунок 4.7 – Схема структурна розгортання для системи обліку продукції під час виробництва

4.3.4 Специфікація функцій

Функції класів програмного забезпечення для RESP API наведені в таблиці 4.1.

Таблиця 4.1 – Функції класів програмного забезпечення для RESP API

Назва	Примітка
validatePassword(login, password)	Дана функція призначена для валідації даних при аутентифікації користувача в системі. Параметрами для неї є 2 значення – логін та пароль користувача. Повертає логічне значення true, якщо логін та пароль введені правильно, та false у разі помилки
getCustomers(limit, sortField, sortType)	Функція призначена для отримання даних про клієнтів системи. Параметрами є 3 значення: limit, що обмежує кількість об’єктів, sortField, що визначає поле за яким відбувається сортування та sortType, що визначає порядок сортування. Ця функція повертає масив JSON об’єктів, кожен з яких містить дані про клієнта системи

Продовження таблиці 4.1

Назва	Примітка
getCustomerById(id)	Функція призначена для отримання даних про певного клієнта системи. Єдиним параметром є id – ідентифікатор клієнта в базі даних. Ця функція повертає JSON об’єкт, який містить дані про шуканого клієнта системи
addCustomer(data)	Функція призначена для створення нового клієнта системи. Параметр data – це JSON об’єкт, з усіма даними про новоствореного клієнта. Ця функція повертає логічне значення true, якщо створення пройшло успішно, і false у разі виникнення помилки
updateCustomer(id, data)	Функція призначена для оновлення даних про певного клієнта системи. Параметр id – це ідентифікатор клієнта, дані якого потрібно переписати, а data – це JSON об’єкт, з новими даними про клієнта. Ця функція повертає логічне значення true, якщо оновлення пройшло успішно, і false у разі виникнення помилки

Продовження таблиці 4.1

Назва	Примітка
deleteCustomer(id)	Функція призначена для видалення даних про певного клієнта системи. Параметр id – це ідентифікатор клієнта, якого необхідно видалити, в базі даних. Ця функція повертає логічне значення true, якщо видалення пройшло успішно, і false у разі виникнення помилки
getOperations(limit, sortField, sortType)	Функція призначена для отримання даних про операції здійснені над продуктами. Параметрами є 3 значення: limit, що обмежує кількість об’єктів, sortField, що визначає поле за яким відбувається сортування та sortType, що визначає порядок сортування. Ця функція повертає масив JSON об’єктів, кожен з яких містить дані про здійснену операцію
getOperationById(id)	Функція призначена для отримання даних про певну здійснену операцію. Єдиним параметром є id – ідентифікатор операції в базі даних. Ця функція повертає JSON об’єкт, який містить дані про шукану операцію над продуктом

Продовження таблиці 4.1

Назва	Примітка
getOperationsByProductId(id)	Функція призначена для отримання даних про операції здійснені над певним продуктом. Єдиним параметром є id – ідентифікатор продукту в базі даних. Ця функція повертає масив JSON об’єктів, кожен з яких містить дані про здійснену операцію
getOperationsByTechnologistId(id)	Функція призначена для отримання даних про операції здійснені певним технологом. Єдиним параметром є id – ідентифікатор технолога в базі даних. Ця функція повертає масив JSON об’єктів, кожен з яких містить дані про здійснену операцію
addOperation(data)	Функція призначена для створення запису про здійснену операцію. Параметр data – це JSON об’єкт, з усіма даними про операцію. Ця функція повертає логічне значення true, якщо створення пройшло успішно, і false у разі виникнення помилки

Продовження таблиці 4.1

Назва	Примітка
getOperationTypes()	Функція призначена для отримання даних про типи операцій на виробництві. Ця функція повертає масив JSON об’єктів, кожен з яких містить дані про певний тип операції
getOperationTypeById(id)	Функція призначена для отримання даних про певний тип операції. Єдиним параметром є id – ідентифікатор типу операції в базі даних. Ця функція повертає JSON об’єкт, який містить дані про шуканий тип операції
addOperationType(data)	Функція призначена для створення нового типу операції. Параметр data – це JSON об’єкт, з усіма даними про тип операції. Ця функція повертає логічне значення true, якщо створення пройшло успішно, і false у разі виникнення помилки

Продовження таблиці 4.1

Назва	Примітка
updateOperationType(id, data)	Функція призначена для оновлення даних про тип операції на виробництві. Параметр id – це ідентифікатор запису про тип операції, дані якого потрібно переписати, а data – це JSON об’єкт, з новими даними цей тип. Ця функція повертає логічне значення true, якщо оновлення пройшло успішно, і false у разі виникнення помилки
deleteOperationType(id)	Функція призначена для видалення даних про певний тип операції. Параметр id – це ідентифікатор типу, який необхідно видалити, в базі даних. Ця функція повертає логічне значення true, якщо видалення пройшло успішно, і false у разі виникнення помилки

Продовження таблиці 4.1

Назва	Примітка
getProducts(limit, sortField, sortType)	Функція призначена для отримання даних про продукцію на виробництві. Параметрами є 3 значення: limit, що обмежує кількість об’єктів, sortField, що визначає поле за яким відбувається сортування та sortType, що визначає порядок сортування. Ця функція повертає масив JSON об’єктів, кожен з яких містить дані про певний продукт
getProductById(id)	Функція призначена для отримання даних про певний продукт на виробництві. Єдиним параметром є id – ідентифікатор продукту в базі даних. Ця функція повертає JSON об’єкт, який містить дані про шуканий продукт
getProductsByCustomerId(id)	Функція призначена для отримання даних про продукцію придбану певним клієнтом. Єдиним параметром є id – ідентифікатор покупця в базі даних. Ця функція повертає масив JSON об’єктів, кожен з яких містить дані про певний продукт

Продовження таблиці 4.1

Назва	Примітка
getProductsByTechnologistId(id)	Функція призначена для отримання даних про продукцію виготовлену певним технологом. Єдиним параметром є id – ідентифікатор технолога в базі даних. Ця функція повертає масив JSON об’єктів, що містять дані про виготовлені вироби
addProduct(data)	Функція призначена для створення запису про новий продукт на виробництві. Параметр data – це JSON об’єкт, з усіма даними про новостворений виріб. Ця функція повертає логічне значення true, якщо створення пройшло успішно, і false у разі виникнення помилки
getProductTypes()	Функція призначена для отримання даних про типи продуктів на виробництві. Ця функція повертає масив JSON об’єктів, кожен з яких містить дані про тип продукції
getProductTypeByName(name)	Функція призначена для отримання даних про певний тип продукції. Єдиним параметром є name – назва типу продукції в базі даних. Ця функція повертає JSON об’єкт, що містить дані про тип продукції

Продовження таблиці 4.1

Назва	Примітка
addProductType(data)	Функція призначена для створення нового типу продукції. Параметр data – це JSON об’єкт, з усіма даними про тип продукції. Ця функція повертає логічне значення true, якщо створення пройшло успішно, і false у разі виникнення помилки
updateProductType(id, data)	Функція призначена для оновлення даних про тип продукції на виробництві. Параметр id – це ідентифікатор запису про тип продукції, дані якого потрібно переписати, а data – це JSON об’єкт, з новими даними цей тип. Ця функція повертає логічне значення true, якщо оновлення пройшло успішно, і false у разі виникнення помилки
deleteProductType(id)	Функція призначена для видалення даних про певний тип продукції. Параметр id – це ідентифікатор типу, який необхідно видалити, в базі даних. Ця функція повертає логічне значення true, якщо видалення пройшло успішно, і false у разі виникнення помилки

Продовження таблиці 4.1

Назва	Примітка
getPurchases(limit, sortField, sortType)	Функція призначена для отримання даних про здійснені покупки. Параметрами є 3 значення: limit, що обмежує кількість об’єктів, sortField, що визначає поле за яким відбувається сортування та sortType, що визначає порядок сортування. Ця функція повертає масив JSON об’єктів, кожен з яких містить дані про певну покупку
getPurchaseById(id)	Функція призначена для отримання даних про певний певну покупку. Єдиним параметром є id – ідентифікатор покупки в базі даних. Ця функція повертає JSON об’єкт, який містить дані про шукану покупку
getPurchasesByCustomerId(id)	Функція призначена для отримання даних про всі покупки здійснені певним клієнтом. Єдиним параметром є id – ідентифікатор покупця в базі даних. Ця функція повертає масив JSON об’єктів, кожен з яких містить дані про певну покупку

Продовження таблиці 4.1

Назва	Примітка
getPurchasesByProductId(id)	Функція призначена для отримання даних про всі покупки здійснені з даним продуктом. Єдиним параметром є id – ідентифікатор продукту в базі даних. Ця функція повертає масив JSON об’єктів, кожен з яких містить дані про певну покупку
addPurchase(data)	Функція призначена для створення запису про нову покупку продукції. Параметр data – це JSON об’єкт, з усіма даними про нову покупку. Ця функція повертає логічне значення true, якщо створення пройшло успішно, і false у разі виникнення помилки
getTechnologists(limit, sortField, sortType)	Функція призначена для отримання даних про технологів на виробництві. Параметрами є 3 значення: limit, що обмежує кількість об’єктів, sortField, що визначає поле за яким відбувається сортування та sortType, що визначає порядок сортування. Ця функція повертає масив JSON об’єктів, кожен з яких містить дані про певного технолога

Продовження таблиці 4.1

Назва	Примітка
getTechnologistById(id)	Функція призначена для отримання даних про певного технолога. Єдиним параметром є id – ідентифікатор технолога в базі даних. Ця функція повертає JSON об’єкт, який містить дані про шуканого технолога
addTechnologist(data)	Функція призначена для створення нового технолога. Параметр data – це JSON об’єкт, з усіма даними про нового технолога. Ця функція повертає логічне значення true, якщо створення пройшло успішно, і false у разі виникнення помилки
updateTechnologist(id, data)	Функція призначена для оновлення даних про певного технолога на виробництві. Параметр id – це ідентифікатор запису про технолога, дані якого потрібно переписати, а data – це JSON об’єкт, з новими даними про нього. Ця функція повертає логічне значення true, якщо оновлення пройшло успішно, і false у разі виникнення помилки

Продовження таблиці 4.1

Назва	Примітка
deleteTechnologist(id)	Функція призначена для видалення даних про технолога. Параметр id – це ідентифікатор технолога, якого необхідно видалити, в базі даних. Ця функція повертає логічне значення true, якщо видалення пройшло успішно, і false у разі виникнення помилки
getTechnologistTypes()	Функція призначена для отримання даних про типи технологів на виробництві. Ця функція повертає масив JSON об’єктів, кожен з яких містить дані про певний тип технолога
getTechnologistTypeByName(name)	Функція призначена для отримання даних про певний тип технолога. Єдиним параметром є name – назва типу технолога в базі даних. Ця функція повертає JSON об’єкт, що містить дані про тип технолога
addTechnologistType(data)	Функція призначена для створення нового типу технолога. Параметр data – це JSON об’єкт, з усіма даними про тип технолога. Ця функція повертає логічне значення true, якщо створення пройшло успішно, і false при помилці

Продовження таблиці 4.1

Назва	Примітка
updateTechnologistType(id, data)	Функція призначена для оновлення даних про тип технолога на виробництві. Параметр id – це ідентифікатор запису про тип технолога, дані якого потрібно переписати, а data – це JSON об’єкт, з новими даними цей тип. Ця функція повертає логічне значення true, якщо оновлення пройшло успішно, і false у разі виникнення помилки
deleteTechnologistType(id)	Функція призначена для видалення даних про певний тип технолога. Параметр id – це ідентифікатор типу, який необхідно видалити, в базі даних. Ця функція повертає логічне значення true, якщо видалення пройшло успішно, і false у разі виникнення помилки

Функції класів програмного забезпечення для мобільного застосунку наведені в таблиці 4.2.

Таблиця 4.2 – Функції класів програмного забезпечення для мобільного застосунку

Назва	**Примітка**
_auth(loginObj)	Дана функція призначена для аутентифікації користувача в системі. Параметром є loginObj – JSON об’єкт з двома полями login та password, який буде передаватись в RESP API. Повертає логічне значення true, якщо автентифікація пройшла успішно, та false у разі помилки
_startNfc()	Функція, що вмикає NFC-модуль на смартфоні. Повертає попередження у випадку, якщо телефон не підтримує NFC
_startDetection()	Функція, яка призначає NFC-модулю розпочати виявлення NFC-мітки.
_onTagDiscovered()	Функція, яка виконується тоді, коли NFC-модуль на смартфоні виявив NFC-мітку
_stopDetection()	Функція, яка призначає NFC-модулю закінчити виявлення NFC-мітки
_goToNfcSetting()	Функція, яка відкриває налаштування NFC на смартфоні. Працює лише для телефонів з операційною системою Android

Продовження таблиці 4.2

Назва	Примітка
_parseText()	Функція, яка розшифровує та повертає текст, який був зчитаний з NFC-мітки
_formatDatetime(str)	Функція, яка форматує дату і час, що записані в рядок str, до необхідного формату
_loadData()	Функція, яка отримує дані з REST API, які необхідні для даного екрану мобільного застосунку. Повертає помилку у випадку, якщо дані неможливо отримати

4.4 Опис звітів

Адміністратор має можливість переглянути статистичний звіт про роботу певного технолога – кількість виготовленої продукції, кількість дефектної продукції, середня швидкість операції, кількість операції виконаних з затримкою.

Також можна переглянути інформацію про певні типи продукції – кількість продаж, середня швидкість виробництва, відсоток бракованих товарів. Доступ до цих звітів має лише адміністратор і переглянути їх він може в приватному обліковому записі адміністратора системи.

Висновок до розділу

В даному розділі було розглянуто технології, за допомогою яких було реалізовано програмний продукт. Також показано два види діаграм із детальним поясненням до них. Розглянуто загальні вимоги для технічних засобів, та було наведено специфікацію до усіх функцій системи.

5 ТЕХНОЛОГІЧНИЙ РОЗДІЛ

5.1 Керівництво користувача

Користуватись нашою системою можна з будь-якого смартфону, тому що програмний продукт представлений стандартним застосунком для Android та IOS.

При запуску програми появляється сторінка авторизації користувача в системі, представлена на рисунку 5.1.

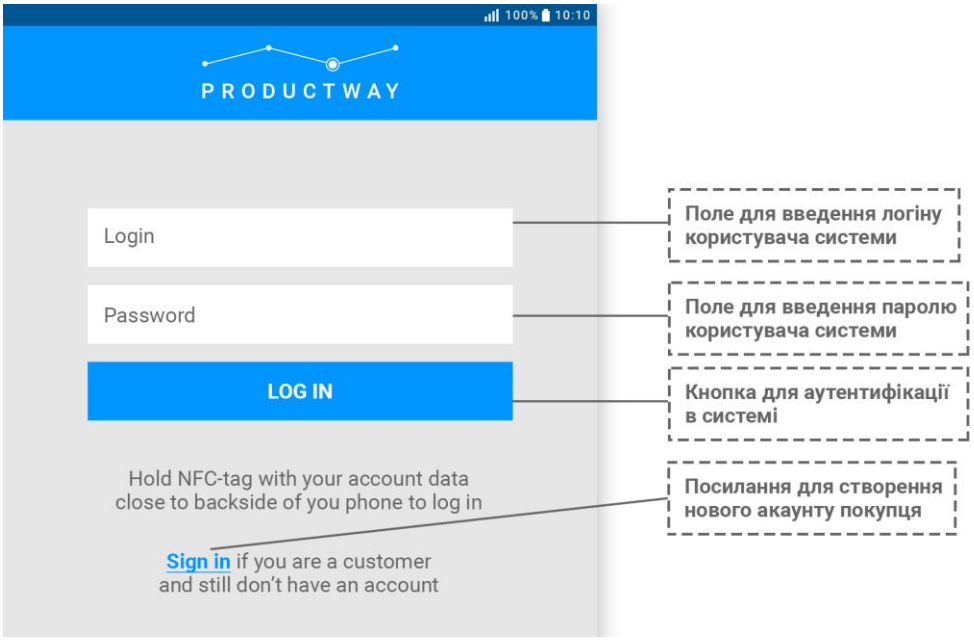

Рисунок 5.1 – Сторінка авторизації користувача в системі

На цій сторінці є 2 поля, в які користувач повинен ввести свої авторизаційні дані, а саме логін і пароль. Також є кнопка «LOG IN» після натискання якої, відбувається авторизація в системі. Технологи та адміністратор можуть здійснювати авторизацію за допомогою NFC-мітки, на яку записані їх логін та пароль. Для цього достатньо лише піднести її до задньої частини телефону і система автоматично прочитає їх, розшифрує і проведе авторизацію. Клієнт, який ще не має власного облікового запису може зареєструватись в системі натиснувши на кнопку «Sign in», після чого він перейде на сторінку реєстрації покупців.

Розглянемо що відбувається коли адміністратор авторизувався в системі. Одразу після цього відкривається сторінка облікового запису адміністратора системи, що зображена на рисунку 5.2.

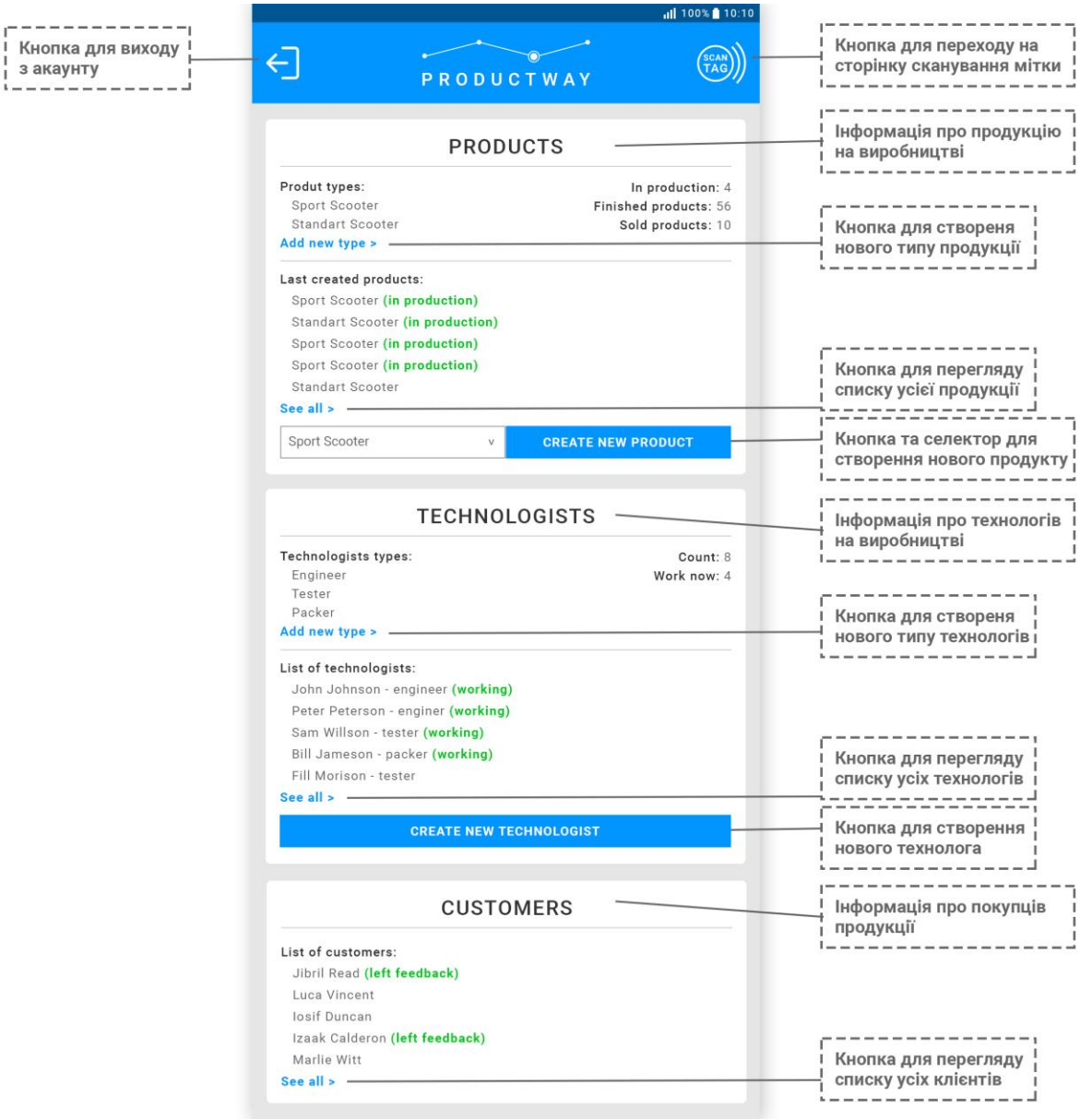

Рисунок 5.2 – Сторінка облікового запису адміністратора системи

На цьому екрані є вся необхідна для адміністратора система про продукцію, технологів та клієнтів системи. В хедері застосунку є 2 кнопки – при натисканні лівої відбувається вихід з облікового запису, при натисканні правої користувач переходить на сторінку сканування NFC-мітки продукту, що зображена на рисунку 5.3.

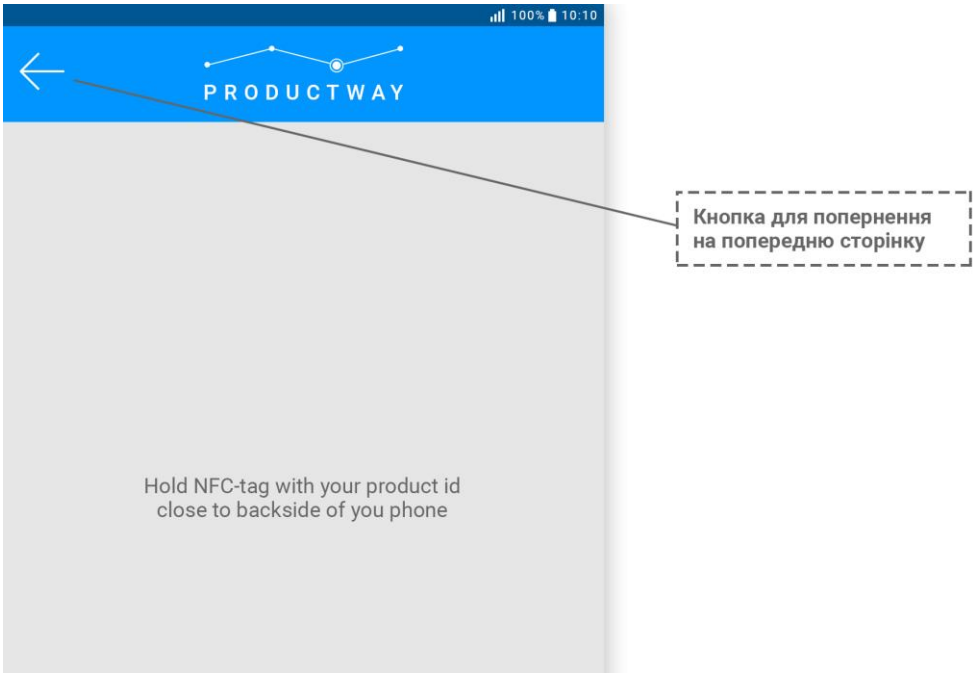

Рисунок 5.3 – Сторінка сканування NFC-мітки продукту

В хедері тепер є кнопка у вигляді стрілки, яка дозволяє повернутись на попередню сторінку. Якщо смартфон не підтримує NFC то система видасть попередження і автоматично закриє дану сторінку. Якщо ж NFC-модуль присутній в телефоні, проте він не є ввімкнений, система запропонує перейти в налаштування і ввімкнути його. Після цього на екрані появляється повідомлення «Піднесіть NFC-мітку до задньої частини вашого телефону» і як тільки мітка буде просканована система відкриє нову сторінку з інформацією про даний продукт, яка зображена на рисунку 5.4. Також її можна відкрити з панелі адміністратора натиснувши на певний запис в списку продуктів.

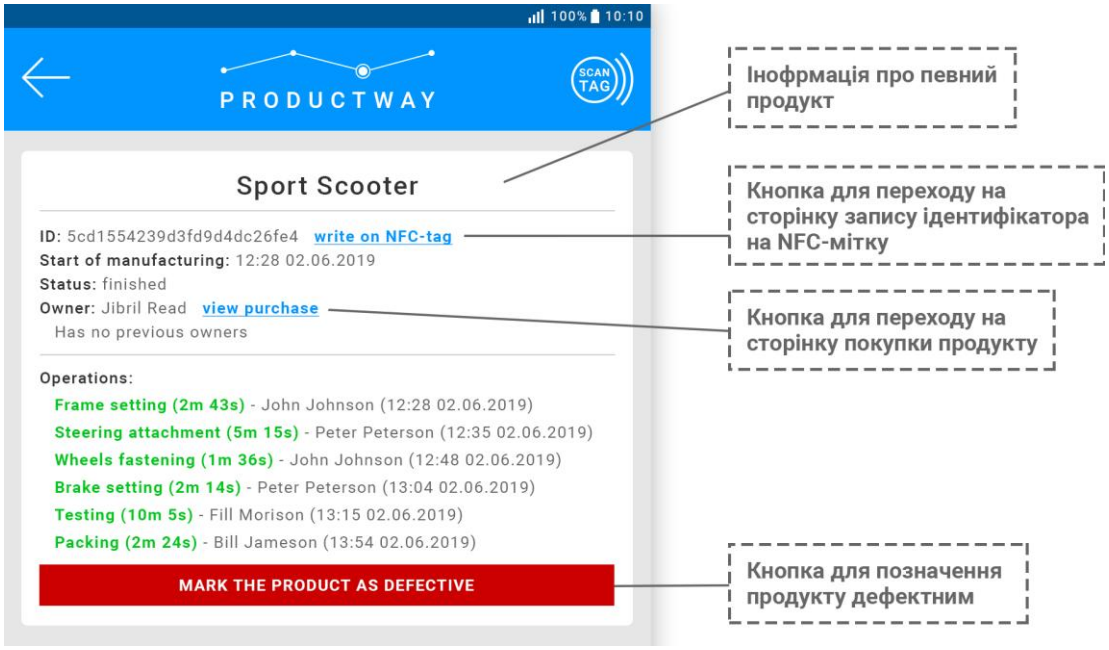

Рисунок 5.4 – Сторінка з даними про продукт для адміністратора

На цій сторінці адміністратор має можливість переглянути всі дані про продукт. При натисканні на кнопку «write on NFC-tag», що знаходиться одразу поруч з ідентифікатором продукту, адміністратор переходить на сторінку, яка аналогічна до сторінки зображеній на рисунку 5.3. Після того як адміністратор піднесе NFC-мітку до телефону, ідентифікатор буде записаний на нього, система видасть сповіщення, про вдале завершення запису і автоматично поверне на попередню сторінку.

За допомогою кнопки «Mark the product as defective» він може позначити, що даний продукт має дефекти після виробництва. Після цього, всі технологи, які були відповідальні за виготовлення цього виробу отримають штраф.

При натисканні на кнопку «View purchase» відбувається перехід на сторінку покупки товару, що зображена на рисунку 5.5.

Рисунок 5.5 – Сторінка з даними про покупку товару для адміністратора

Тут адміністратор може переглянути інформацію про покупку, а також відгук від покупця. Нижче є поле, в якому адміністратор може залишити свою відповідь для покупця, натиснувши кнопку «Send Answer» після заповнення.

Якщо на сторінці облікового запису адміністратора натиснути на назву типу продукції, то відкривається сторінка редагування, яка виглядає аналогічно до сторінки створення нового типу. На рисунку 5.6 можна побачити її вигляд.

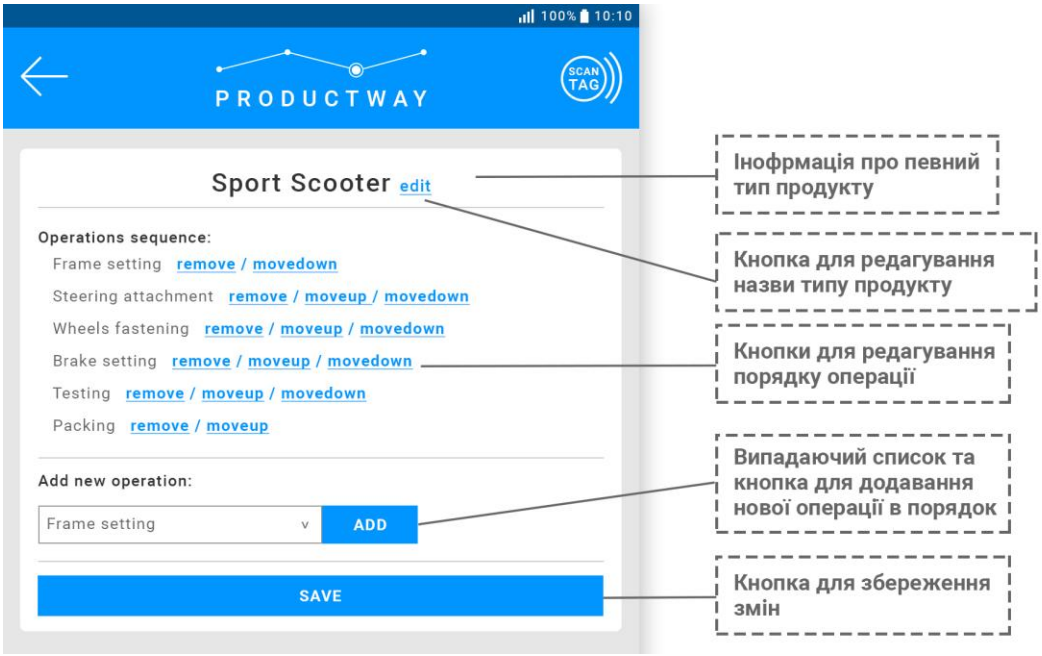

Рисунок 5.6 – Сторінка редагування/створення типу продукції

Кнопкою «Edit» адміністратор може редагувати назву цього типу. Нижче, за допомогою кнопок «remove», «moveup», «movedown» можна змінювати порядок операцій необхідний для виготовлення даного виробу. Якщо ж потрібно додати нову операцію, то нижче є випадаючий список з усіма операціями і кнопка «ADD», якою можна підтвердити додавання даної операції в чергу. Кнопка «SAVE» дозволяє завершити редагування і зберігає усі внесені зміни в базу даних.

На сторінці адміністратора є селектор з типами продукції та кнопка «CREATE NEW PRODUCT», які відповідають за створення нового запису про продукт, тим самим видаючи замовлення для технологів. При виборі необхідного типу, і натисканні кнопки, відкривається сторінка даного продукту, такого ж типу, як зображена на рисунку 5.4.

Нижче розміщений блок, що надає всю інформацію про технологів на виробництві. Якщо натиснути на назву певного типу технолога, то відкриється сторінка з інформацією, яка зображена на рисунку 5.7. При натисканні на кнопку «add new type» відкривається аналогічна сторінка тільки без існуючих даних.

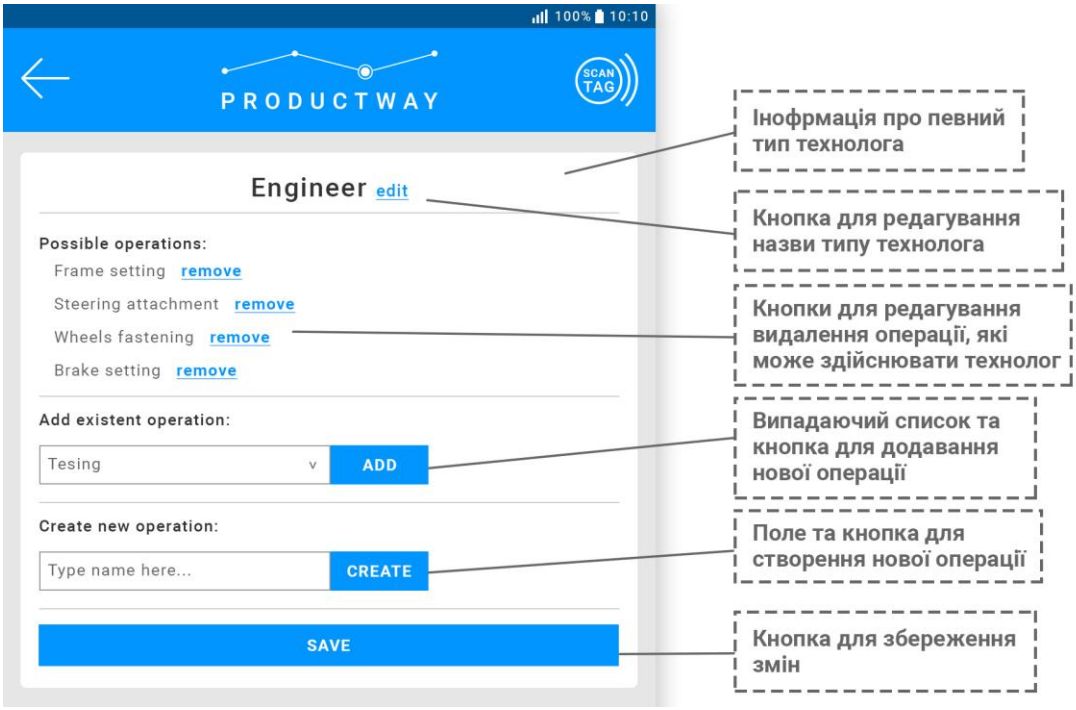

Рисунок 5.6 – Сторінка редагування/створення типу технолога

Кнопкою «Edit» адміністратор може редагувати назву цього типу. Нижче, за допомогою кнопок «remove» можна змінювати список операцій, які може здійснювати даний тип технологів. Якщо ж потрібно додати нову операцію, то нижче є випадаючий список з усіма операціями і кнопка «ADD», якою можна підтвердити додавання даної операції в чергу. Нижче є поле і кнопка «CREATE», за допомогою яких можна створити нову операцію, яка ще не була записана в базу даних, і додати її в список. Кнопка «SAVE» дозволяє завершити редагування і зберігає усі внесені зміни в базу даних.

При натисканні на певного технолога на сторінці адміністратора, користувач переходить на сторінку, що зображена на рисунку 5.7. На ній можна переглянути всі дані про цього працівника, а також відредагувати їх за допомогою кнопок «edit». Також при натисканні кнопки «write auth-data on NFC-tag» адміністратор переходить на сторінку, яка аналогічна до сторінки зображеній на рисунку 5.3. Тут адміністратор може записати авторизаційні дані технолога на NFC-мітку, які потім можна видати даному працівнику. Для цього необхідно піднести NFC-мітку до телефону і після цього система

видасть сповіщення, про вдале завершення запису і автоматично поверне на попередню сторінку.

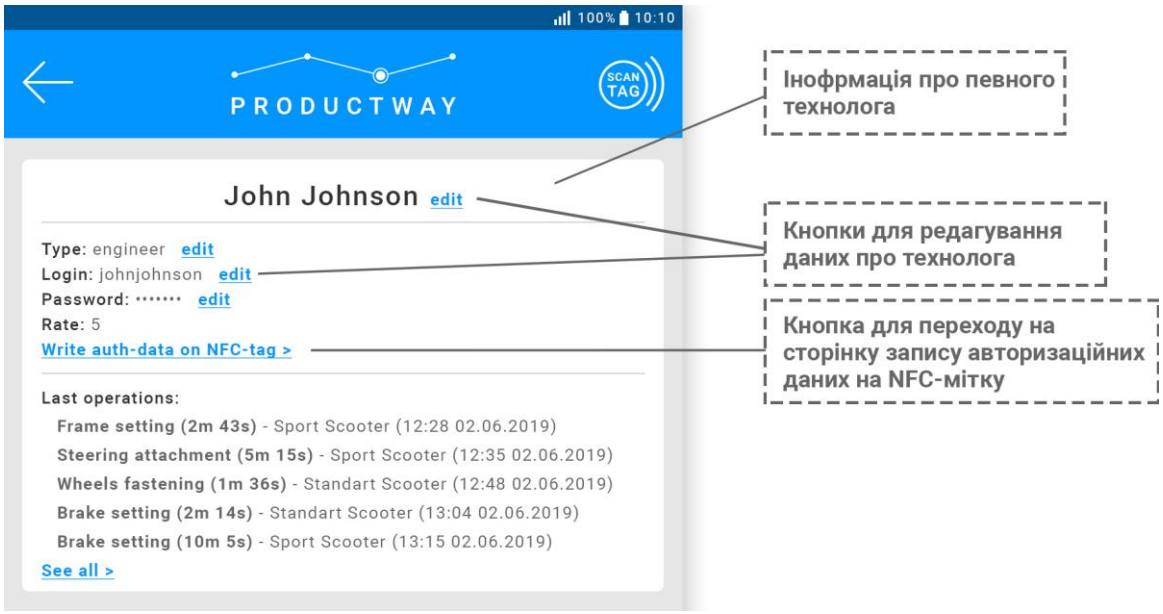

Рисунок 5.7 – Сторінка редагування/створення технолога

На цю ж сторінку відбувається перехід при натисканні кнопки «CREATE NEW TECHNOLOGIST» на сторінці облікового запису адміністратора, проте дані будуть відсутні.

При натисканні на певного клієнта на сторінці адміністратора, користувач переходить на сторінку, що зображена на рисунку 5.8. На ній можна переглянути всі дані про цього покупця, а також перейти на сторінки покупок, що вже були зображені на рисунку 5.5.

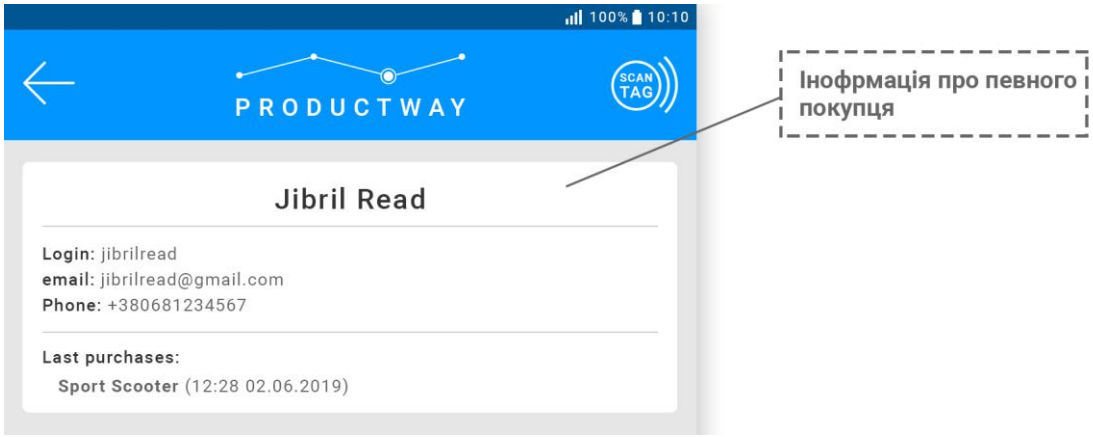

Рисунок 5.8 – Сторінка з даними про клієнта для адміністратора

Коли авторизацію здійснює технолог, то він бачить сторінку свого облікового запису, що зображена на рисунку 5.9.

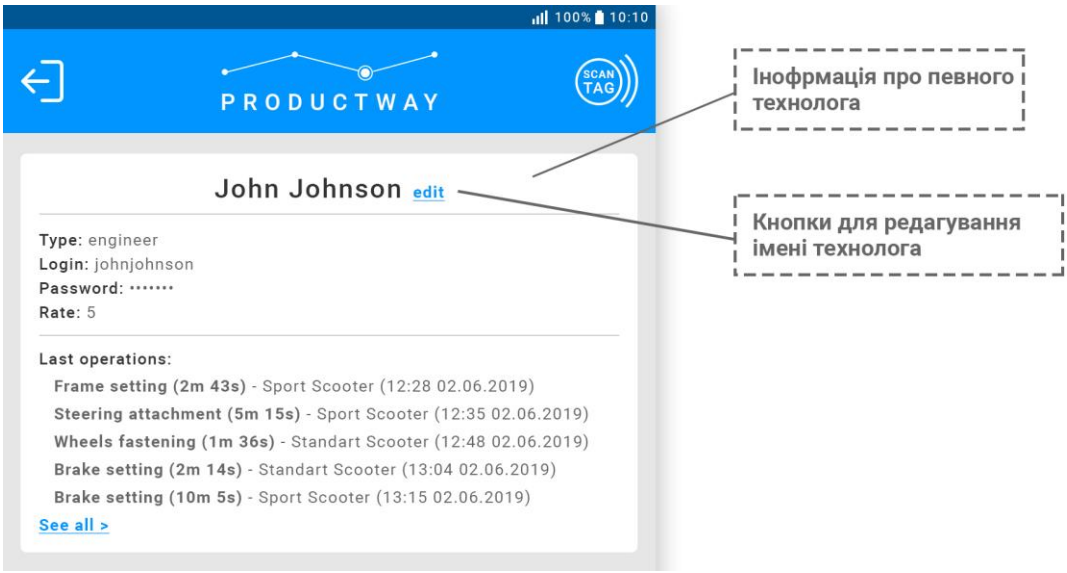

Рисунок 5.9 – Сторінка облікового запису технолога системи

Технолог може переглянути інформацію про себе, проте змінити може лише ім’я, оскільки всі інші дані задає адміністратор. Також нижче є список останній здійснених операцій даним технологом. При натисканні на тип продукту над яким вона була здійснена відкривається сторінка про продукцію. При скануванні NFC-мітки продукту також відкривається сторінка про даний продукт, яка зображена на рисунку 5.10.

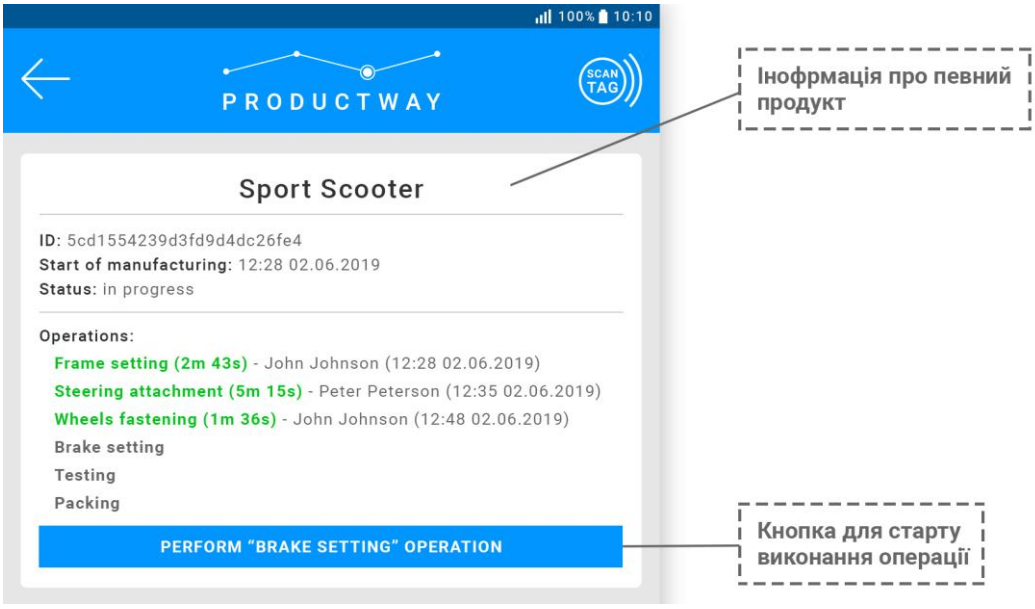

Рисунок 5.10 – Сторінка з даними про продукт для технолога

На цій сторінці є вся інформація про продукт, та операції, які вже були здійснені, і які необхідно здійснити. Якщо не всі операції виконані, то нижче буде кнопка «PERFORM OPERATION», при натисканні на яку, технолог записує в систему, те що саме він здійснює наступну операції.

Коли авторизацію здійснює покупець, то він бачить сторінку свого облікового запису, що зображена на рисунку 5.11.

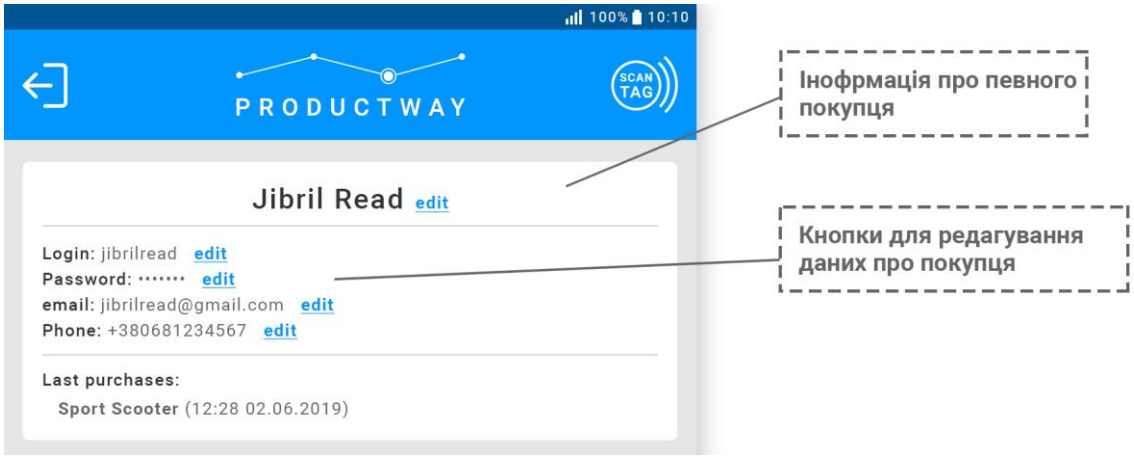

Рисунок 5.11 – Сторінка облікового запису покупця системи

Тут покупець може редагувати дані свого облікового запису а також переглянути всі свої покупки. При натисканні на назву купленого продукту, або при скануванні NFC-мітки з ідентифікатором продукту відкривається сторінка, що зображена на рисунку 5.12.

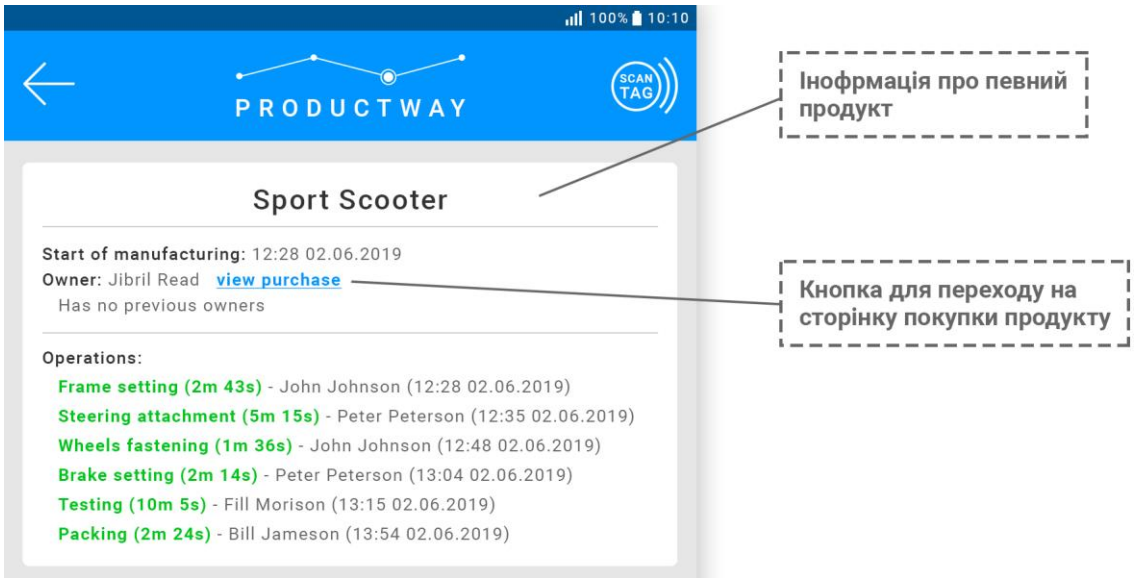

Рисунок 5.12 – Сторінка з даними про продукт для покупця

При натисканні на кнопку «view purchase» система переходить на сторінку покупки, де клієнт може залишити відгук про товар. Ця сторінка зображена на рисунку 5.13.

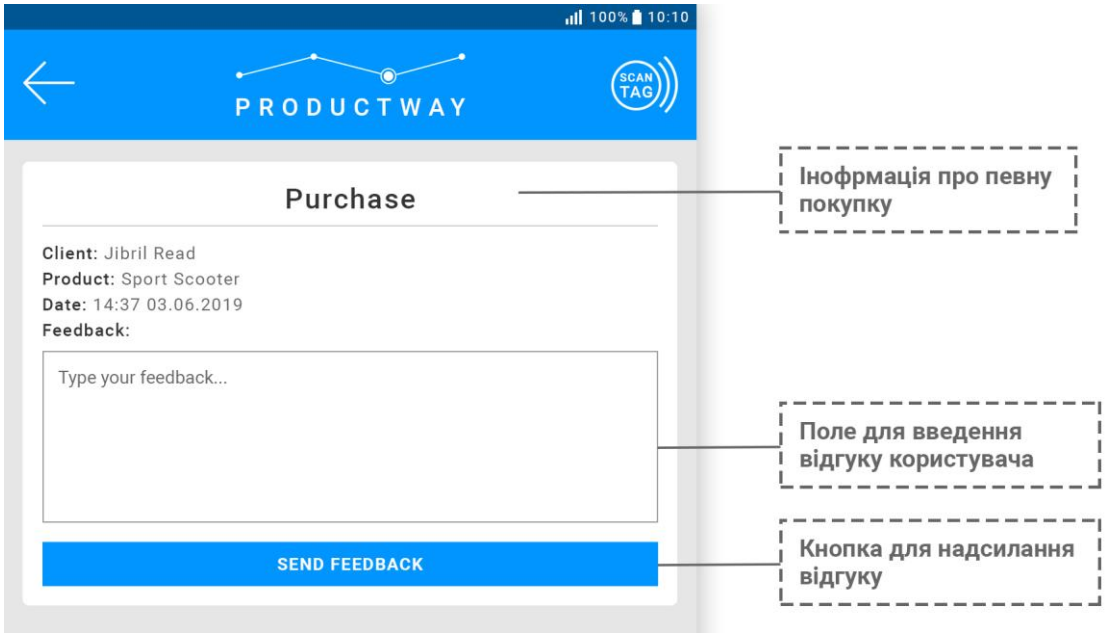

Рисунок 5.13 – Сторінка з даними про покупку для клієнта

5.2 Випробування програмного продукту

5.2.1 Мета випробувань

Метою випробувань являється перевірка відповідності функцій комплексу задач обліку продукції за допомогою NFC-міток вимогам технічного завдання.

5.2.2 Загальні положення

Випробування проводяться на основі наступних документів:

– ГОСТ 34.603–92. Інформаційна технологія. Види випробувань автоматизованих систем;
– ГОСТ РД 50-34.698-90. Автоматизовані системи вимог до змісту документів.

5.2.3 Результати випробувань

Розглянемо основні функції системи та перевіримо їх відповідність вимогам технічного завдання. У таблицях 5.1 – 5.8 наведено перелік випробувань основних функціональних можливостей.

Таблиця 5.1 – Перевірка можливості авторизації за допомогою NFC-мітки

Початковий стан	Відкрита сторінка авторизації
Схема проведення тесту	Піднести NFC-мітку з авторизаційними даними
Очікуваний результат	Відкриється сторінка з обліковим записом користувача, чиї авторизаційні дані були записані на мітку
Стан після випробування	Авторизація здійснена, відкрита сторінка користувача

Таблиця 5.2 – Перевірка можливості відкриття сторінки продукту за допомогою сканування NFC-мітки з його ідентифікатором

Початковий стан	Відкрита сторінка сканування NFC-мітки
Схема проведення тесту	Піднести NFC-мітку з ідентифікатором продукту
Очікуваний результат	Відкриється сторінка з даними продукту, ідентифікатор якого був записаний на мітку
Стан після випробування	Зчитування успішне, відкрита сторінка продукту

Таблиця 5.3 – Перевірка можливості запису авторизаційних даних користувача на NFC-мітку

Початковий стан	Відкрита сторінка з даними про технолога з облікового запису адміністратора
Схема проведення тесту	Натиснути кнопку «write auth-data on NFC-tag» і тоді піднести NFC-мітку до телефону
Очікуваний результат	Дані будуть записані на мітку, система видасть сповіщення про успішний запис і повернеться на попередній екран
Стан після випробування	Записування успішне, відкрита сторінка з даними про технолога

Таблиця 5.4 – Перевірка можливості запису ідентифікатора продукту на NFC-мітку

Початковий стан	Відкрита сторінка з даними про продукт з облікового запису адміністратора
Схема проведення тесту	Натиснути кнопку «write ID on NFC-tag» і тоді піднести NFC-мітку до телефону
Очікуваний результат	Дані будуть записані на мітку, система видасть сповіщення про успішний запис і повернеться на попередній екран
Стан після випробування	Записування успішне, відкрита сторінка з даними про продукт

Таблиця 5.5 – Перевірка отримання штрафу технологами за бракований продукт

Початковий стан	Відкрита сторінка з даними про продукт з облікового запису адміністратора
Схема проведення тесту	Натиснути кнопку «mark the product as defective»
Очікуваний результат	На сторінці технологів, які проводили операції над цим продуктом поле «rank» збільшиться на одиницю
Стан після випробування	Значення поля «rank» збільшилось у всіх технологів, що працювали над виробництвом даного продукту

Таблиця 5.6 – Перевірка створення запису про здійснення операції технологом над продуктом

Початковий стан	Відкрита сторінка з даними про продукт з акакунту технолога
Схема проведення тесту	Натиснути кнопку «perform operation» і після цього кнопку «finish performing»
Очікуваний результат	Проведена операція в списку засвітиться зеленим, а біля неї буде ім’я технолога, з облікового запису якого була здійснена операція
Стан після випробування	Запис операції отримав зелений колір, і біля нього появилось ім’я технолога

Таблиця 5.7 – Перевірка надсилання покупцем відгуку про придбаний товар

Початковий стан	Відкрита сторінка з даними про покупку з акакунту покупця
Схема проведення тесту	Заповнити поле для відгуку та натиснути кнопку «send feedback»
Очікуваний результат	Надісланий відгук збережеться у поле «feedback» на сторінці про покупку, а зі сторінки адміністратора появиться можливість надіслати відповідь
Стан після випробування	Текст відгуку записаний в поле «feedback», з облікового запису адміністратора є можливість надіслати відповідь

Таблиця 5.8 – Перевірка надсилання адміністратором відповіді на надісланий користувачем відгук

Початковий стан	Відкрита сторінка з даними про покупку з акакунту адміністратора
Схема проведення тесту	Заповнити поле для відповіді та натиснути кнопку «send answer»
Очікуваний результат	Надісланий відгук збережеться у поле «Administrator's answer» на сторінці про покупку
Стан після випробування	Текст відгуку записаний в поле «Administrator's answer», з облікового запису покупця є можливість прочитати його

Висновок до розділу

В даному розділі було сформоване керівництво користувача системи, де було визначено дії, що може здійснювати користувачі в розробленій нами системі, а також наведені випробування реалізованого програмного продукту. Було описано мету випробувань, загальні положення та наведено результати цих випробувань.

Також були описані результати проведених випробувань щодо функціональності нашої системи. Випробування проводилися шляхом функціонального тестування. Було описано основні функції системи та перевірено їх відповідність вимогам технічного завдання. У результаті проведення випробувань було виправлено всі виявлені помилки.

ЗАГАЛЬНІ ВИСНОВКИ

У ході виконання дипломного проекту детально розглянуті питання, які виникають в процесі обліку продукції на підприємстві. Були виділені основні ключові етапи, притаманні процесу, та взаємозв'язки між ними, сформовано предметне середовище, визначено та описано процеси діяльності, а також розглянуті наявні аналоги системи. Крім того, була сформована функціональна модель системи - описані користувачі системи, їх ролі та можливі дії у системі.

Була досліджена технологія NFC, спосіб функціонування та аналоги технології що використовуються в подібних системах. Також були переглянуті методи користування, можливості застосування, переваги та недоліки використання.

На основі даних, отриманих в процесі аналізу, було сформульовано завдання та продумана структура системи обліку продукції. Для збереження даних було вирішено використовувати REST API на віддаленому сервері, а для користувачів – розробити мобільний застосунок, оскільки майже кожен смартфон зараз має вбудований NFC-модуль.

Для розробки REST API були використані технології Node.js і фреймворк Express.js, та документо-орієнтована система керування базами даних MongoDB для збереження даних.

Для розробки застосунку була використана технологія React Native, що дозволяє використовувати мову JavaScript, для розробки кросплатформених мобільних застосунків. Тобто ця технологія транслює JS-код одразу в 2 види застосунків – ті що підтримуються Android системами, і ті що працюють на IOS.

Наведена детальна інструкція користувача по експлуатації комплексу задач, описана методика проведення випробувань, яка показує можливість введення програми в експлуатацію.

ПЕРЕЛІК ПОСИЛАНЬ

1. History of the Barcode Scanner [Електронний ресурс]. – Режим доступу: https://www.dbk.com/resources/barcode-scanner-history.html
2. The pros and cons of QR codes [Електронний ресурс]. – Режим доступу: https://econsultancy.com/the-pros-and-cons-of-qr-codes/
3. Using NFC For Asset Tracking & Inventory Management [Електронний ресурс]. – Режим доступу: http://www.ahg.com/business-mobile-apps-blog/nfc-for-asset-inventory-tracking.html
4. What is NFC & how does it work? [Електронний ресурс]. – Режим доступу: https://www.androidauthority.com/what-is-nfc-270730/
5. Система ShelfAware RFID для быстрого учета движения продукции на складе? [Електронний ресурс]. – Режим доступу: http://nfcukraine.com/shelfaware-rfid-system-for-fast-tracking-of-goods-movement-in-the-warehouse/
6. Comparison of RFID, NFC and Barcode for Inventory Tracking [Електронний ресурс]. – Режим доступу: https://rfid4u.com/comparison-of-rfid-nfc-and-barcode-for-inventory-tracking-part-2-nfc-barcode/
7. What Is MD5? (MD5 Message-Digest Algorithm) [Електронний ресурс]. – Режим доступу: https://www.lifewire.com/what-is-md5-2625937
8. Засаливание паролей [Електронний ресурс]. – Режим доступу: https://habr.com/ru/post/39079/
9. Хэш-функция MD5 [Електронний ресурс]. – Режим доступу: https://habr.com/ru/sandbox/26876/
10. Learning React Native [Електронний ресурс]. – Режим доступу: oreilly.com/library/view/learning-react-native/9781491929049/ch01.html
11. Що таке RESTful API? [Електронний ресурс]. – Режим доступу: https://codeguida.com/post/601